POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Natural Language Processing techniques for epidemics monitoring

Supervisors

Prof. Giuseppe RIZZO

Sara DE LUCA

Juan José MÁRQUEZ VILLACÍS

Candidate

Giacomo ROSSO

07 2024

Summary

The ever-increasing and publicly accessible flow of news represents a valuable resource for extracting information to detect health crises using machine learning techniques. The primary objective of this thesis is to accurately classify news articles by topic, with a focus on those related to health, and subsequently categorise them by specific diseases. This research aims to explore innovative architectures and techniques in Natural Language Processing to analyse and classify the news stream. In addition, a robust test suite will be developed to evaluate the reliability of the developed models. The resulting classifications will generate time series of news related to specific diseases. This thesis is part of a much larger project in collaboration with LINKS Foundation called TrustAlert, which aims at early detection of health crises.

Acknowledgements

ACKNOWLEDGMENTS

"Alla mia famiglia, che mi ha sempre supportato e sopportato."

Table of Contents

List of Tables

List of Figures

Acronyms

AI

Artificial Intelligence

API

Application Programming Interface

BERT

Bidirectional Encoder Representations from Transformers

BoW

Bag of Words

ER

Entity Relationship

FAISS

Facebook AI Similarity Search

FFNN

Feed Forward Neural Network

GDELT

Global Database of Events, Language, and Tone

GDPR

General Data Protection Regulation

GKG

Global Knowledge Graph

GPT

Generative Pre-trained Transformer

GQG

Global Quotations Graph

ICD9

International Classification of Diseases, 9th revision

ICU

Intensive Care Unit

IPTC

International Press Telecommunications Council

LLM

Large Language Modeling

LSTM

Long Short Term Memory

MIMIC

Medical Information Mart for Intensive Care

MLM

Masked Language Modeling

MLP

Multi Layer Perceptron

NLP

Natural Language Processing

RAG

Retrieval Augmented Generation

RNN

Recurrent Neural Networks

SQL

Structured Query Language

SVM

Support Vector Machines

TF-DF

Term Frequency - Document Frequency

TF-IDF

Term Frequency - Inverse Document Frequency

Chapter 1

Introduction

1.1 Aim of this thesis project

This thesis work fit within the LINKS foundation's contribution to TrustAlert project. Specifically, this work investigates Natural Language Processing techniques to filter and classify online news retrieved from an online news stream called GDELT. This process is done in a completely unsupervised scenario. During this thesis a test suite is developed to quantitatively evaluate the performances of the developed model responsible for filtering and classification.

The main contribution of this thesis is the development of an artificial intelligence architecture able to perform first a filtering phase, dividing news related to health topics from the others, then a tagging phase, where the filtered news are tagged with the disease they are reporting on.

Due to the evolving nature of the diseases and the completely unsupervised nature of the available data, a major focus is placed on the zero shot application of the models involved. Thus, the focus is on exploration of models that use different pre-training strategies, data quality and format rather than the traditional fine-tuning of these models.

Finally, a labelled dataset has been developed for this thesis and fine-tuned versions of these models have been developed through a strategy based on the contrastive learning instead of traditional fine-tuning based on classification heads. A foundamental step was to formalize the problem not as a multi-label classification one but as a retrieval problem using the notion of semantic search. This intuition was the first step towards the concept of contrastive learning originally developed for computer vision tasks.

1.2 What is TrustAlert?

TrustAlert is a project founded by Fondazione Compagnia di San Paolo and developed by the cooperation between different parties: Department of Clinical and Biological Sciences of University of Torino, Department of Informatics of University of Torino, Foundation Bruno Kessler and Links Foundation.

The TrustAlert main goals are:

1. **Intercept and understand the healthcare crisis** by extracting information from healthcare, clinical data and news reporting on communicable diseases.

2. **Assess the impact** by cross-referencing healthcare data and classifying population health and vulnerability.

3. **Provide containment actions** by conducting simulation to optimize resource allocation.

Envisioning the TrustAlert project, it is possible to identify different parts that work together, and the LINKS Foundation is responsible for the "early detection" part. This means developing a tool able to automatically detect anomalies in the news streams and the healthcare or hospital data. The most important data sources for early detection are primarily text data in the form of online news streams such as GDELT as well as healthcare data and clinical data provided by hospitals and healthcare facilities. All this data needs to be analyzed and classified, and the best choice for performing these operations is the use of Large Language Models (LLM). This Master thesis projects focus on the news classification part of TrustAlert early detection phase. Specifically, by investigating NLP techniques and developing a model able to filter and tag news with the diseases they are reporting on.

Given the nature of the data handled, it is also foundamental that TrustAlert project aligns its core values with European values by using algorithms and databases that are compliant with the GDPR and the AI Act. This ensures the protection of the privacy of data subjects and the centrality of the human being throughout the development of the project.

1.3 Challenges

One of the biggest challenges is working with completely unstructured and unlabeled text: from freely written news to hospital records and discharge notes. This means that no fixed structure is imposed on the text data and that there are often no meaningful labels associated with the text in the datasets related to healthcare news currently available.

Another important challenge is the nature of Large Language Models, which are resource-intensive and data-hungry. Therefore, there is the need to find a way to not train the models involved from scratch every time, but to utilise the zero-shot capabilities of these huge pre-trained models and find a way to tailor the tasks to fully exploit their potential.

1.4 Preview of results

To summarize the value brought by this work:

- Development of a flexible tagger capable of classifying any text with any set of labels using the zero-shot capabilities of its freely chosen, pre-trained backbone. This model achieves results of over 98% in Accuracy@1 in the first filtering phase and over 90% in the disease tagging phase.
- Development of a labelled dataset that can be used to quantitatively evaluate the tagger's performance on a text format and style similar to the one of the news the model will actually filter. These datasets are easily replicable and updatable using reliable data sources such as BBC and World Health Organization.
- Comparison between the zero-shot application of base models and their fine-tuned versions on specific domains such as the ICD9 codes classification. The model that best performed on this thesis test was the SentenceTransformer with an MPNET backbone.
- Investigation of contrastive learning techniques and Masked Language Modeling techniques to fine-tune LLM models for classification without losing their zero-shot capabilities. Starting from a state-of-the-art model known as MPNET, it is finetuned through contrastive learning techniques to be adapted to ICD9 tagging task.

Chapter 2

Background

This chapter presents the problem from a top-level perspective and explains possible machine learning and deep learning techniques, with a particular focus on natural language processing, that can be used to address the main challenges of this thesis considering also the constraints imposed by TrustAlert.

In today's information-rich world, the early detection of health crises is a complex yet critical task. The sheer volume of data generated from news sources, social media and healthcare facilities presents a significant challenge in monitoring and responding to potential epidemiological threats. This is where the TrustAlert project comes in. The main focus of the project, led by the LINKS Foundation, is the early detection component. This involves the development of tools to classify and analyse text data from sources such as the Global Database of Events, Language, and Tone (GDELT).

One of the main challenges in this project is dealing with completely unstructured and unlabelled text. This includes freely written news articles and hospital records, which require sophisticated techniques for effective classification and analysis. The aim of this chapter is to provide a brief introduction to NLP and all the techniques and architectures used in the methods (chapter 4) and experiments (chapter 5). This includes a look at techniques for representing text in data structures that can be processed by machine learning models, the history of such models, from *Word2Vec* to *Transformers* and other *Seq2Seq* models, and finally a presentation of the different techniques used to train these models effectively.

2.1 An introduction over Natural Language Processing

Natural Language Processing is a prominent branch of artificial intelligence that focuses on the interaction between computers and human language representing a crucial intersection of computer science, linguistics and artificial intelligence. It enables machines to understand, interpret and generate human language, opening up numerous applications in transformative industries such as the TrustAlert project in the healthcare industry. In the context of this thesis, NLP techniques are central to transforming unstructured and unlabelled text from news articles and hospital records into structured, analysable information.

The history of NLP evolution can be drawn as a line from from simple *Bag of Words* and *TF-IDF* to dense, context-aware embedding like *Word2Vec*, *GloVe*, and *Transformers* (*BERT*, *GPT*). In the meanwhile the architectures transitioned from traditional Machine Learning models (*Naive Bayes*, *SVM*) to advanced neural networks (*RNN*, *LSTM*) and *Transformer* architectures.

2.2 NLP techniques

The first step in NLP involves converting text into data structures that machine learning models can process. This process, known as *text representation* or *text vectorization*, has evolved significantly over the years, but essentially can be defined as obtaining a **numeric high dimensional vector** from a textual input.

2.2.1 Sparse Vector Representation

The first data structure used to represent textual data was to generate a global vocabulary and represent each document as a binary vector of terms present in each document. Thus, a representation similar to a one hot encoding

	sentence	meows	dog	cat	animals	has	are	and	legs	four
0	dog and cat are animals	0	1	1	1	0	1	1	0	0
1	dog has four legs	0	1	0	0	1	0	0	1	1
2	cat meows	1	0	1	0	0	0	0	0	0

Figure 2.1: Example of one hot encoding of textual data, in this case sentences.

Other possible representations that leverages once more the frequentistic approach towards words are *TF-IDF* and *TF-DF*. These techniques compute for

each word the times it appears in the current document weighted by its overall appearances in the whole collection. The *TF-IDF* penalizes words that are spread over all documents, suitable for heterogeneous collections. The *TF-DF* rewards word that are spread over all documents, suitable for homogeneous collections.

	sentence	meows	dog	cat	animals	has	are	and	legs	four
0	dog and cat are animals	0.000000	0.373022	0.373022	0.490479	0.000000	0.490479	0.490479	0.000000	0.000000
1	dog has four legs	0.000000	0.402040	0.000000	0.000000	0.528635	0.000000	0.000000	0.528635	0.528635
2	cat meows	0.795961	0.000000	0.605349	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Figure 2.2: Example of TF-IDF encoding of sentences.

The mathematical formulation of TF-IDF and TF-DF are the following:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$
$$\text{TF-DF}(t, d) = \text{TF}(t, d) \times \text{DF}(t)$$

Notice that t is a word term, d is a document or sentence and D is the collection of documents. Each term is defined as:

$$\text{TF}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

$$\text{IDF}(t) = \log\left(\frac{N}{|\{d \in D : t \in d\}|}\right)$$

$$\text{DF}(t) = |\{d \in D : t \in d\}|$$

However useful, these textual features are computed on the frequencies of occurrences of the main textual units (words or char n-grams). Therefore this approach is suitable only for syntactic embedding, because no semantic relation or meaning is preserved, and small vocabularies, due to the resulting dimension of the embedding being the vocabulary size itself.

2.2.2 Dense Vector Representations

Representation learning aims to learn embeddings for words, sentences and documents through a self-supervised process on the raw input data. Each input is then mapped to a dense vector in a space where the distance between vectors is related to their semantic distance. This approach is less interpretable because meaning is no longer localised in a feature, but is spread across the vector representation. Furthermore, studying the latent space can provide new insights into the hidden

information in the text. All embedding models start from the so-called ***"distributional hypothesis"*** [1]: when a target word is chosen, its semantic meaning can be inferred from nearby words. In other words, the context in which a word is used defines the meaning of the word itself.

Word2Vec

One of the first architectures proposed, with two different training techniques, is ***Word2Vec*** [2]. It builds directly upon the *"distributional hypothesis"* by analyzing a large document corpus and outputting a static embedding for every encountered word. This is done using a *Feedforward Neural Net Language Model* [3] and other simplier architectures with different training strategies like *CBOW* and *Skip-Gram*, as shown in Fig 2.3.

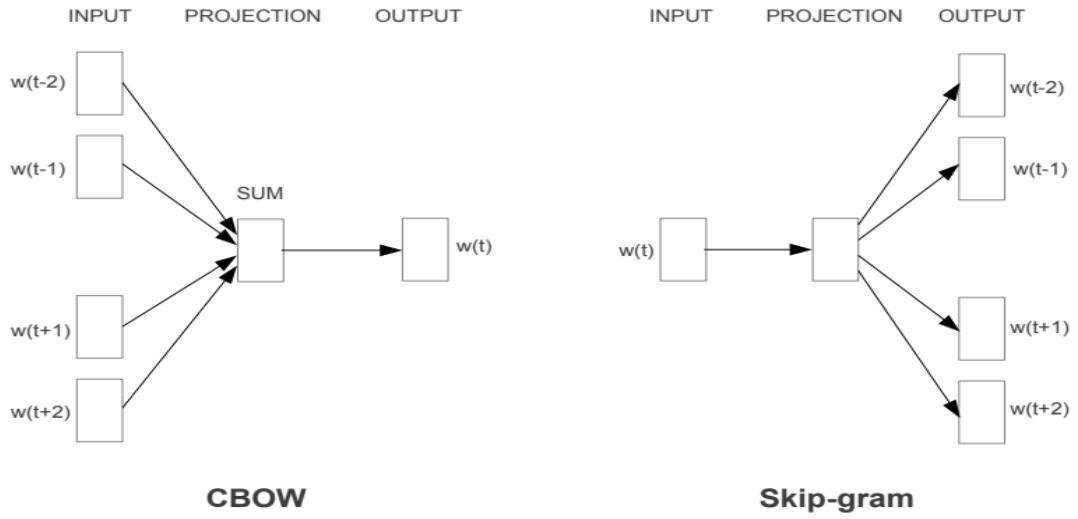

Figure 2.3: The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word. Figure from [2].

As specified before, this methods and all the other presented in this section produce "static embedding". This means that each word is mapped to the same embedding each time is encountered. This behaviour, even if simple to understand and implement, does not take into account the fact that the same word can have different meaning. To generate more advanced embedding deep learning techniques are required, which are presented in section 2.3.

FastText

One of the main problems of *Word2Vec* is the impossibility to have embedding for out of vocabulary words. In other words, if a word was never seen during the training phase, it has no learnt representation. ***FastText*** [4] [5] makes a first step towards solving this issue by producing embedding for both whole words and their sub-words, called *n-grams* (pieces of n consecutive textual units like characters).

Figure 2.4: FastText sub-word representation using the word "eating" as example.

The final embedding of the word is represented as the sum of its own n-gram embedding and the embedding of the special sequence represented by the whole word. In this way, each word can have a representation enriched with sub-word information, and for those that were never seen during training, their representation is generated using only their n-grams. The sub-word concept, introduced in several works other than *FastText*, such as the adaptation of *Byte Pair Encoding* [6] to textual data [7] and *WordPiece* [8] algorithm, was seminal for tokenisation, which is the basis of modern NLP [9].

Figure 2.5: FastText training resembles closely the contrastive learning framework.

2.3 Deep NLP architectures

A significant advancement in NLP occurred with the development of architectures capable of processing sequences of textual data while capturing both short and long distance dependencies. These architectures typically feature numerous learnable parameters and are composed of sequences of layers or repeated stacked blocks. These blocks perform operations on the input text to transform it into embedding vectors within a high-dimensional latent space.

Recurrent Neural Networks

A limitation of a Feed Forward Neural Network [10] is that its input must be of fixed length. ***Recurrent Neural Networks*** are flexible in that regard. In fact, unlike FFNNs, RNNs can handle sequences of varying lengths because they process data one element at a time, maintaining a hidden state that captures information about previous elements in the sequence.

As shown in Fig 2.6, RNN can be thought as a cell with a set of "weight, biases and activation functions" (U, W) but also with a "feedback loop" (V) so that $output_{t-1}$ contributes to $output_t$. The hidden, intermediate state is represented by h. The shared weights (U, W, V) allow the RNN to apply the same transformations at each time step, which makes them well-suited for handling sequential data.

To better visualize it one can unfold the cell into sequential steps. In Fig. 2.6 all the weights are shared (U, W, V) and in doing so the RNN can have sequences of variable length as input. Therefore, the data is fed sequentially and as consequence the order of data matters during learning.

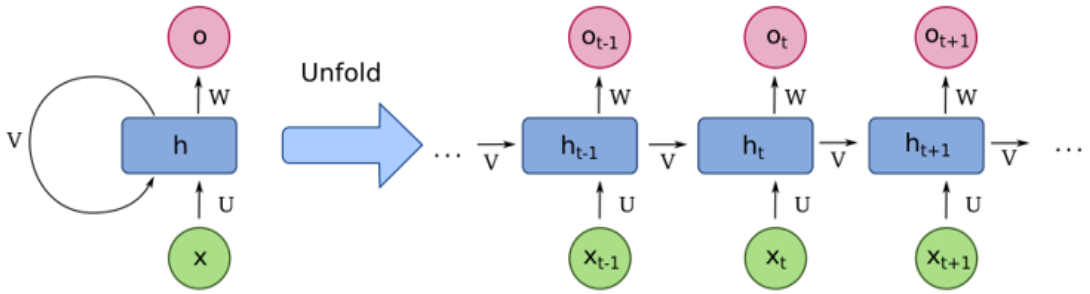

Figure 2.6: RNN architecture. **Left**: the real architecture. **Right**: the unfolding of the architecture as sequential data x_t is fed to the RNN.

Two of the main problems of RNNs are the *Gradient Vanishing* and *Gradient Explosion* for longer sequences because of the excessive length of the back propagation update along the feedback loop. To specifically solve these problems the LSTM were introduced.

Long Short Term Memory

The evolution of the *RNN* to avoid gradient update problems is the ***Long Short Term Memory*** architecture [11], shown in Fig 2.7.

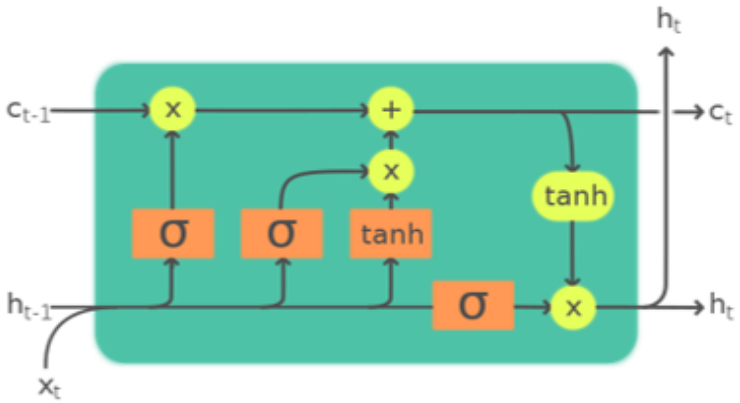

Figure 2.7: LSTM cell architecture. It has three different inputs: "Sequential input" x_t, "Short Term Memory" h_{t-1} and "Long Term Memory" c_{t-1}. It produces two outputs: "Updated Short Memory" h_t and "Updated Long Memory" c_t.

The LSTM cell uses two different activation functions in for two different reasons:

- **Sigmoid** $\sigma : \mathcal{R} \rightarrow [0,1]$: to compute % of retention of a quantity.
- **Hyperbolic Tangent** $tanh : \mathcal{R} \rightarrow [-1,1]$: to compute a new value scaled by the result of the sigmoid activation function.

The functioning of the LSTM cell involves the following steps:

1. **Retention of Long-Term Memory** c_{t-1}: The cell retains a percentage of the long-term memory from the previous time step c_{t-1}, modulated by the sequential input x_t and the short-term memory (hidden state) from the previous time step h_{t-1} through a sigmoid activation function σ. This is referred to as ***Forget Gate***.
2. **Computation of Potential Contribution to Long-Term Memory**: The cell computes a potential contribution to the long-term memory using the activation function $\tanh(x_t)$. It then decides the percentage of this contribution to add to the retained long-term memory to obtain the updated long-term memory c_t. This is referred to as ***Cell state update***.
3. **Updating Short-Term Memory** h_t: The cell uses the updated long-term memory c_t and the sequential input x_t to update the short-term memory h_t. This is referred to as ***Output Gate***.

ELMo: "Embeddings from Language Models"

By utilizing gates, LSTMs effectively regulate the flow of information, ensuring that critical signals are retained and not diminished over time. This mechanism helps preserve gradients during backpropagation, enabling the architecture to handle longer sequences of data. This pivotal architectural advancement led to the creation of the first models capable of generating deep contextualized embedding, such as **ELMo** a.k.a. "**Embeddings from Language Models**" [12]. These embedding capture both syntactic and semantic aspects of words, accounting for variations in word meanings across different linguistic contexts.

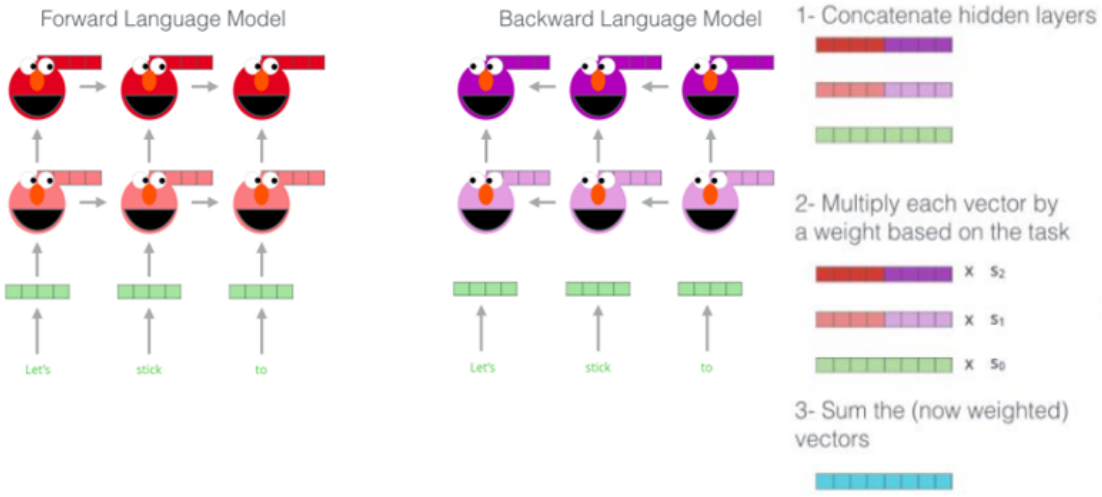

Figure 2.8: Elmo architecture. Multi layer deep bidirectional LSTM.

The ELMo architecture is based on a **deep bidirectional LSTM trained on a language modeling task**, meaning it predicts the next token in the sequence both left-to-right and right-to-left. This dual perspective enhances the model's understanding of context and improves its ability to capture intricate dependencies within the text. ELMo representations differ from previous architectures in that each representation is a function of the entire input sequence. This is achieved by combining its representations across multiple layers linearly (with learnt weights s_i), as illustrated in Fig 2.8. Each layer captures different levels of abstraction, from basic syntactic information to more nuanced semantic features. In fact, higher-level LSTM states capture context-dependent aspects of word meaning while lower level states model aspects of syntax [12]. While LSTMs introduced critical technical advancements, ELMo brought forward essential concepts that are now prevalent in modern NLP. These concepts include the significance of deep architectures and their ability to model complex relationships within sequences, as well as the importance of pretraining and bidirectionality in processing input sequences.

Sequence to Sequence framework

Taking a step back, one of the main frameworks of problems in NLP are ***"Sequence to Sequence"*** or ***Seq2Seq*** problems. This framework can be described from a top level perspective as having an input sequence of variable length and the need to map it to a correct (*"Machine Translation"*) or meaningful (*"Question Answering"* or *"Summarisation"*) output sequence also of variable length. A generic example is visualized in Fig 2.9.

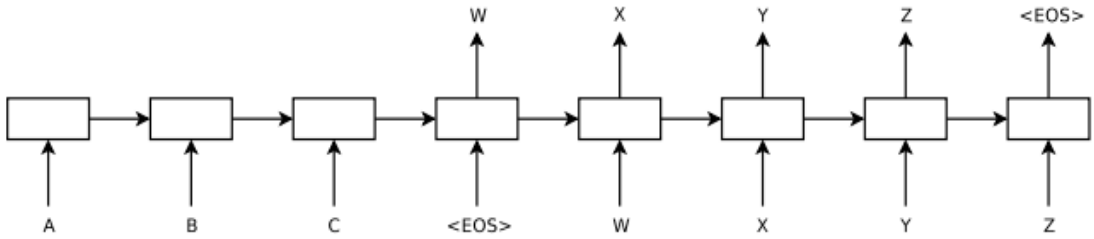

Figure 2.9: Example of Seq2Seq problem. Mapping ABC input sequence to WXYZ output sequence.

Historically, one of the main ideas driving architecture design in NLP was the ***Encoder-Decoder*** structure. This approach was originally inspired by the ***"Continuous Space Language Models"*** or ***CSLM*** [13] [14]. In these models, a fixed length sequence of C words are first represented by *"1-of-n coding"*, where the *i-th* word in the vocabulary is coded by setting the *i-th* element of the vector to 1 and all other elements to 0, and then are projected together, sharing a weight matrix P, and concatenated into a continuous vector space. Lastly, this representation is passed through a non linear hidden linear layer, not dissimilar from what happens in a *FFNN*, to be mapped to a vector containing the posterior probabilities of all words in the vocabulary. The model is trained to maximize the probability of the following word in the sequence.

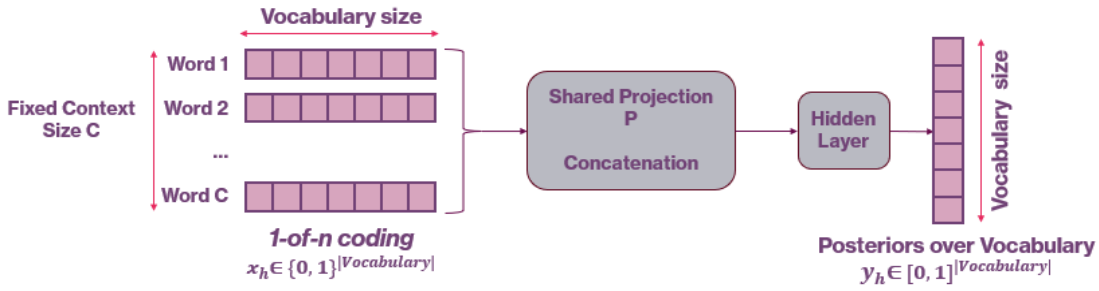

Figure 2.10: Continuous Space Language Model functional diagram.

Sequences, however, pose a challenge for traditional Neural Networks architectures like *FFNNs*. These architectures require that the dimensionality of the inputs and outputs is known and fixed, and this is clearly not guaranteed in a general textual sequence. To address limitations posed by the *FFNNs* and to be able to handle variable length sequences, the ***Encoder-Decoder*** structure was created, using *RNNs* [15] or LSTMs [16] for the inner work of both the Encoder and Decoder parts. As shown in Fig 2.11, the encoder extracts a fixed-length vector representation, the *"context vector" C* from a variable-length input sentence, then the decoder takes C as its initial state and generates a variable-length target sequence.

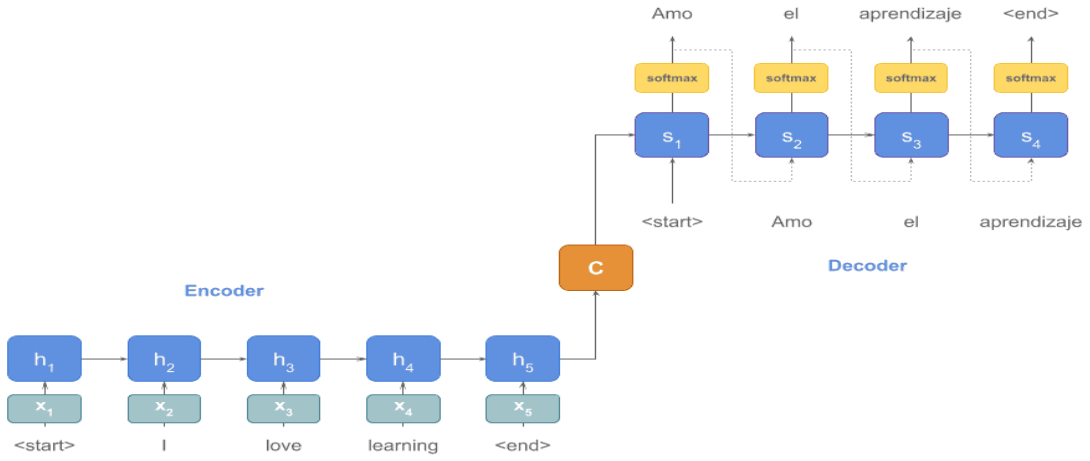

Figure 2.11: Encoder Decoder architecture. The blocks inside both Encoder and Decoder are the visual representation of the unrolling of a RNN or LSTM.

Transformers and "Attention is all you need"

A significant challenge with the encoder-decoder approach is that the encoder must compress all the necessary information from the source sequence into a fixed-length vector. This often results in a rapid decline in performance as the input sentence length increases, due to the loss of detailed information in longer sequences [17].

The most substantial advancement in NLP and sequence analysis in recent years is undoubtedly the development of the ***Attention mechanism*** [18]. This mechanism allows the model to focus on specific parts of the input sequence while generating the output. First introduced to improve machine translation performance with recurrent LSTM encoder-decoder structures [19], the attention mechanism has quickly become a standard in NLP. It has led to the creation of the highly effective ***Transformer*** models [20], which are at the forefront of modern NLP architectures.

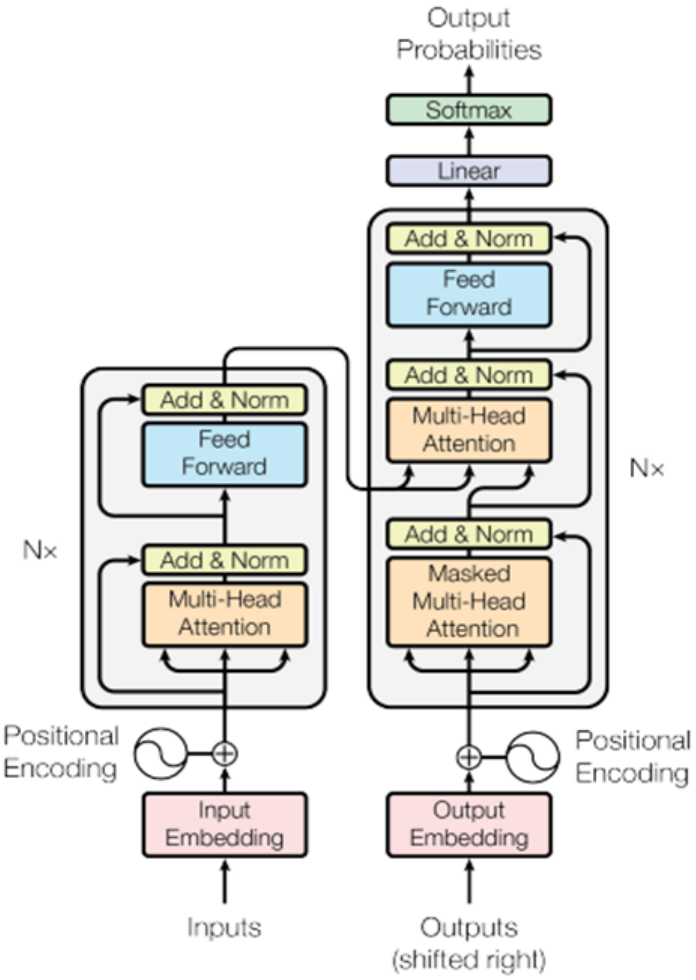

Figure 2.12: Transformer architecture. **Left**: the stack of self-attention layers that behaves like an Encoder. **Right**: the stack of self-attention and cross-attention layers that behaves like a Decoder. Figure from [20].

Transformers [20] are ***auto-regressive encoder-decoder models*** [21] that discard the recurrent inner structure and instead use multiple layers of self-attention and *FFNN*s, resulting in better parallelization and handling of long-range dependencies.

The attention block implemented in the Transformer is defined by three different set of weights (Q_w, K_w, V_w). These weights are used to generate intermediate representations for each token within the input sequence called *Query*s Q, *Key*s K and *Value*s V. These representations allow the model to dynamically weight the importance of each input token.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V$$

In both the encoder and decoder components of the Transformer architecture, the *self-attention* mechanism generates queries Q, keys K, and values V from the input tokens of the sequence. In contrast, during the decoder's *cross-attention* phase, the queries Q are derived from the previous decoder layer, while the keys K and values V are obtained from the encoder's output. To preserve the auto-regressive property, the decoder's input tokens are restricted to attending only to the tokens that have already been generated, thereby preventing any look-ahead bias or foreshadowing.

The absence of recurrent structures eliminates the need for sequential processing of input data, allowing **parallelization** and thus significantly reducing training time. However, there are some drawbacks.

The first minor drawback is that the order of the tokens of the sequences does not play a role during training, and therefore a positional embedding becomes mandatory to manually provide the model with this information. The ***positional encoding*** [22] is designed to provide information about the relative and absolute positions of tokens in the sequence. Moreover, by using only sine and cosine functions, Transformer architecture ensures that the overhead is minimal and easily manageable within the larger computational framework of the model.

Another and more problematic drawback is the **quadratic complexity with respect to sequence length in computing attention**. In fact, when defining attention as in the Transformer architecture, each token computes its similarity to every other token in the sequence, so the quadratic complexity is $\mathcal{O}\Big(\textit{Sequence length}\Big)$. Different attention mechanism have been proposed to overcome this issue such as the ***Longformer*** [23] with its local windowed attention mixed with a global attention or ***BigBird*** [24] with its block attention mechanism of sparse, local and global attention.

The last main drawback is that without recurrent structures, now the model has a hard constraint of maximum sequence length it can handle. This is an hyperparameter chosen at train time and can impede wanted behaviour at inference time. Methods to handle this limitations divide into more efficient attention mechanisms (*Longformer*, *BigBird*) or techniques to split the sequence applying relative positional encoding and segment recurrence mechanism such as in **Transformer-XL** [25] and in **XLNet** [26].

BERT and transformer-like models

The promising results and the easy parallelization brought by the Transformer architecture led to an increasing effort in training Language Models able to better capture semantic relationships within the text sequences. The standard behaviour is becoming pretraining Transformer-like architectures on Language Modeling task over a large textual corpus [27] to then fine tune this foundation model on different

possible downstream tasks.

The most influential model in this regard is **BERT: Bidirectional Encoder Representations from Transformers** [9]. The architecture of BERT is a **deep bidirectional Encoder-only Transformer**, which means it retains only the Encoder part of the original Transformer architecture and in its self-attention mechanism each token can attend to both tokens to its left and to its right.

A key concept in modern NLP is the distinction between *pre-training* and *fine-tuning* [27]. Typically, large language models undergo an initial phase of unsupervised or self-supervised training on a large corpus of text to then be adapted to specific tasks in supervised scenarios, as shown with BERT in Fig 2.13.

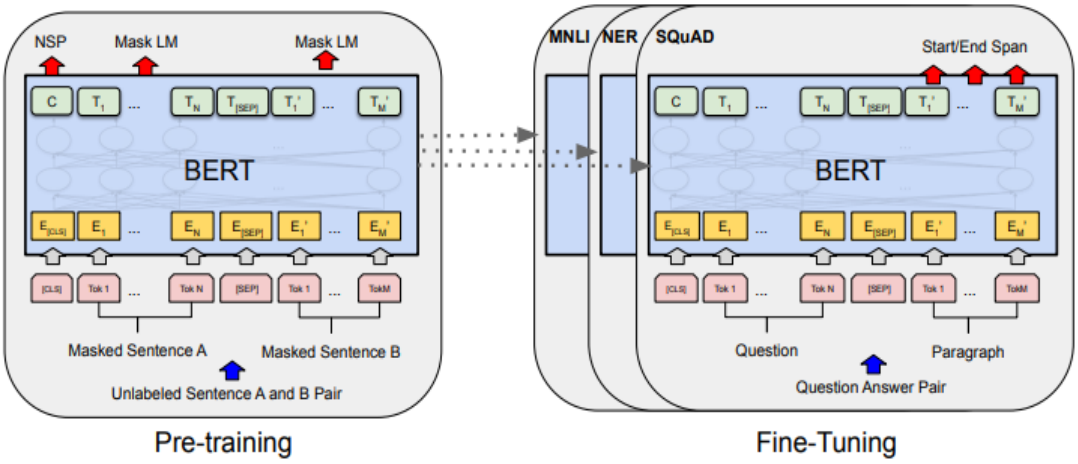

Figure 2.13: BERT is pretrained on a large corpus of text through Masked Language Modeling and Next Sentence Prediction. Then, one model per downstream task is initialized as BERT pretrained and then finetuned on its own specific task. Figure from [9].

The first phase focuses on tasks such as ***Language Modeling*** [27], where the model predicts the next word given the previous words as context, or ***Masked Language Modeling*** [9], where the model attempts to predict words within a text that have been replaced by a special token like *"[MASK]"*, or ***Next Sentence Prediction*** [9], where the model has to predict if two sentences are entangled or not. This first phase is called **pre-training** and provides the model with information about the language distribution.

After pre-training, the model is slightly adapted for the specific task, usually by adding a classification or regression head (a variation of *FFNN*s and *MLP*s) that takes as input the embedding generated by the *pretrained LLM* and provides the task specific output. This second phase is typically supervised, is known as **fine-tuning** and its goal is adapting *LLM*'s general knowledge to specific tasks.

SentenceBERT and how to use effectively BERT Embedding

Being BERT an *Encoder-Only Transformer*, it is a natural choice for extracting a representations of words and sentences in a latent space. However, the architecture of BERT is a token-by-token architecture, therefore each input token has its own embedding. In lots of cases the token representation is not as useful as a sentence or document representation. Some examples may be large-scale semantic similarity comparison, clustering and information retrieval via semantic search, as in this thesis project.

To tackle these kind of problems, *BERT* utilizes a *cross-encoder* approach where two sentences are passed through the Transformer network, and a target value is predicted by a finetuned head. However, this setup becomes impractical for various pair regression tasks due to the large number of possible sentence combinations.

To address this limitation, a common method for clustering and semantic search is to map each sentence to a vector space where semantically similar sentences are positioned close to each other. The most common approach involves either averaging the *BERT* output layer [28] or using the output of the first token (the special token *"[CLS]"*). However, these initial methods produced rather poor sentence embedding [29], often performing worse than older techniques like averaging *GloVe* embedding [30].

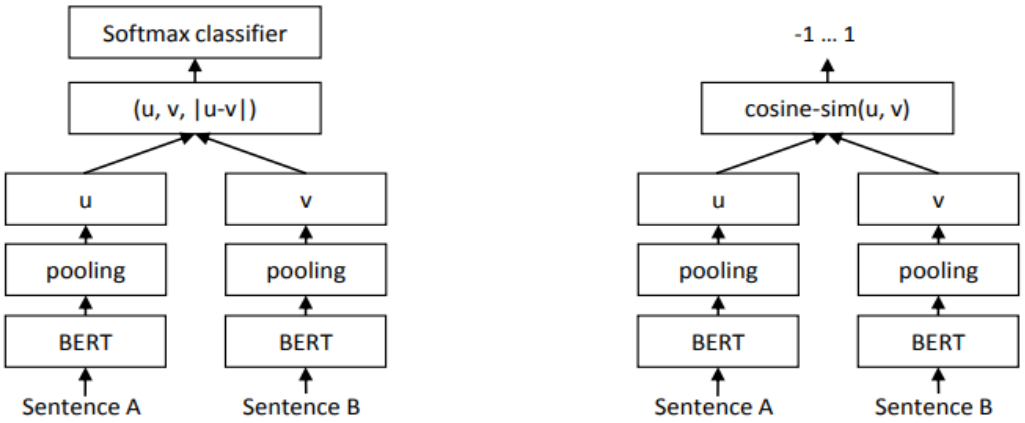

Figure 2.14: SentenceBERT is initialized as BERT and adds pooling operations along with custom objective functions to generate whole-sentence representations. Figure from [29].

The first significant improvement in this area came with ***SBERT: Sentence-BERT*** [29]. This architecture, as shown in Fig 2.14 uses a *siamese network* [31] to generate fixed-sized vectors for input sentences. By finetuning BERT through the

addition of pooling layers and contrastive learning techniques employing similarity measures like *cosine similarity* or *Manhattan/Euclidean distance*, *SBERT* can efficiently identify semantically similar sentences. These similarity measures can be performed extremely efficiently on modern hardware, making SBERT suitable for semantic similarity search and clustering tasks.

BioBERT, domain specific LLM and Vocabularies

Once large language models based on Encoder-only architectures, such as *BERT*, or Decoder-only architectures, such as *GPT-2* [32], became the state of the art in general purpose language understanding and processing, the challenge of handling domain-specific languages and semantics emerged. Researchers began exploring ways to specialize these models for specific domains and, given the scope of this thesis, the most notable example was ***BioBERT*** [33].

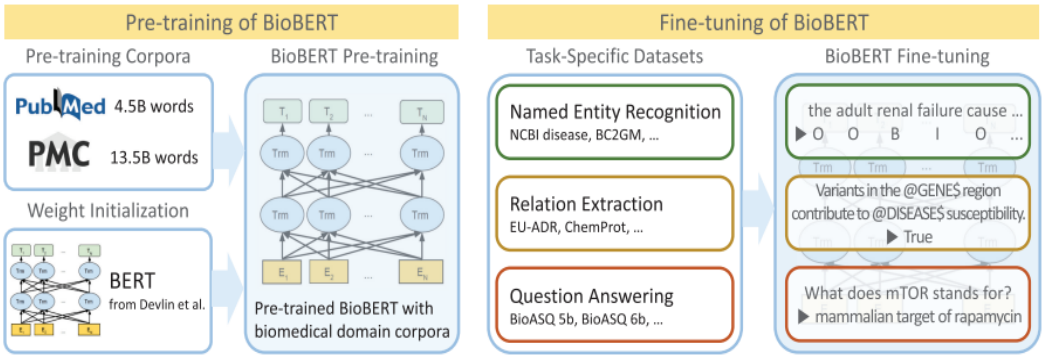

Figure 2.15: BioBERT is initialized as BERT and the same pretraining strategy is applied on a domain specific text collection. Figure from [33].

BioBERT is the successful attempt to specialize a BERT-like model on the biomedical semantic field, where specific terminology is quite common and general purpose *LLM*s have difficulty in achieving performances as high as expected. The strategy adopted, shown in Fig 2.15, was to initialize the pretrained model of BERT that achieved state of the art performances and, maintaining the exact same architecture and vocabulary (*BioBERT-base*), perform an ***"additional pretraining"*** with a domain specific text collections such as PubMed abstracts and PubMed Central full-text articles.

A crucial aspect of *LLM*s is that to fed the text sequences to the Transformer architecture, they first have to represent words as numeric vectors. To do so, they first have to split the sequences into predetermined tokens. This sequence and word sectioning part is achieved through two main components: a tokenizer and a vocabulary. Specifically, the tokenizer split the sequences using the tokens available

in the vocabulary and the initial embedding layer of the Transformer assigns a starting vector representation to each token.

As can be seen, vocabulary plays a crucial role in shaping the results of Transformer. There are two main approaches in the modern literature. The first is **keeping the same vocabulary as the pre-trained model** to be able to use pre-trained weights, thus taking advantage of the expensive pre-training done by others, as in *BioBERT* [33] with *BERT* initialization. The second strategy is to **generate a new domain-specific vocabulary** by Byte Pair Encoding [6] or WordPiece [8] and train from scratch one of the models that achieve top domain performance, thus having a potentially more knowledgeable model but paying the cost of pre-training, as in *SciBERT* [34].

XLNet and the importance of pre-training tasks

Focusing specifically on *Transformer* pretraining tasks, in literature exist mainly two successful approaches: **autoregressive language modeling** and **autoencoding** tasks.

Autoregressive language modeling is a technique where the model predicts the next token in a sequence based on the preceding tokens. It generates text in a left-to-right or right-to-left manner, using the previously generated tokens as context to predict the subsequent ones. This method is foundational in models like *GPT*, which build coherent sequences by continuously expanding from the initial input.

Autoencoding, in contrast, does not focus on explicit density estimation but aims to reconstruct original data from a corrupted input. Some prominent example are *BERT*'s *Masked Language Modeling* or ***BART*** *Denoising* [35]. In MLM, some tokens in the input sequence are replaced with a special symbol, *"[MASK]"*, and the model is trained to recover the original tokens from this corrupted version. Because density estimation is not part of its objective, *BERT* can utilize bidirectional contexts for reconstruction. However, the *"[MASK]"* tokens used during pretraining do not appear in real data during fine-tuning, leading to a pretrain-finetune discrepancy. Additionally, since the masked tokens are not visible in the input, *BERT* cannot model the joint probability using the product rule as autoregressive language models do. This means *BERT* assumes the predicted tokens are independent of each other given the unmasked tokens, which is a significant simplification.

XLNet [26] is the first approach that tries to address the problems of *BERT* pretraining strategy. It tries to solve the Independence assumption of MLM by generalizing the autoregressive language modeling introducing the ***permutation language modeling***. This strategy maximizes the expected log likelihood of a sequence with respect to all possible permutations of the factorization order (or in expectation by sampling some factorization orders).

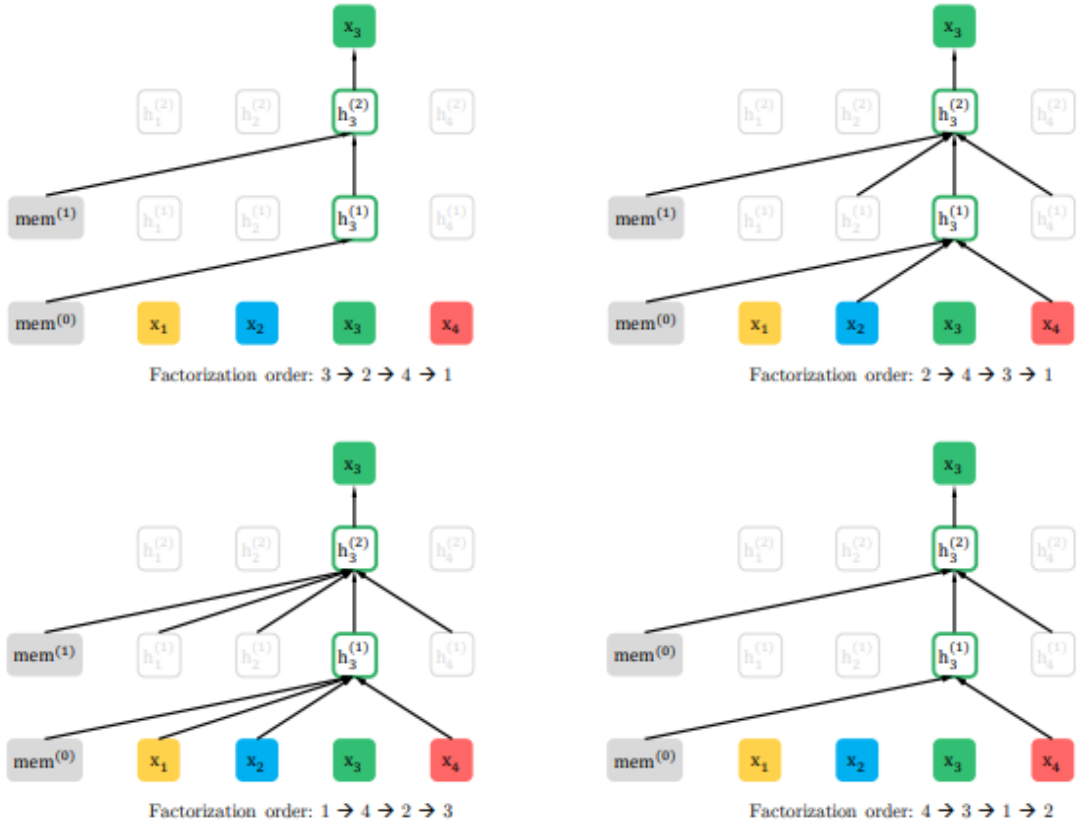

Figure 2.16: Different permuted factorization orders in the sequence [x_1, x_2, x_3, x_4] let the autoregressive pretraining task on x_3 attend to both left-side and right-side tokens. Figure from [26].

As shown in Fig 2.16, given the token x_3 to predict, instead of relying on the previous tokens (x_2 and x_1) we sample some factorization orders and attend to all the previous tokens w.r.t that factorization order. By using permutation, the context for each token can include tokens from both the left and the right, thereby addressing the lack of bidirectionality in traditional autoregressive models without enforcing the independency assumption. Note that the original *Transformer*'s positional embedding is still applied before the permutation, which has the sole purpose of being applied throughout the pretraining to enhance the autoregressive language modeling strategy.

One of the challenges with permuted sequences is preventing the model from seeing the token it is supposed to predict, which would trivialize the task. To do so, a change is needed in what the model is able see, namely the *attention mask*. This mask effectively tells the model for each token what to attend to and what to ignore. By working on those the XLNet is able to enanche the autoregressive

pretraining without trivializing the task.

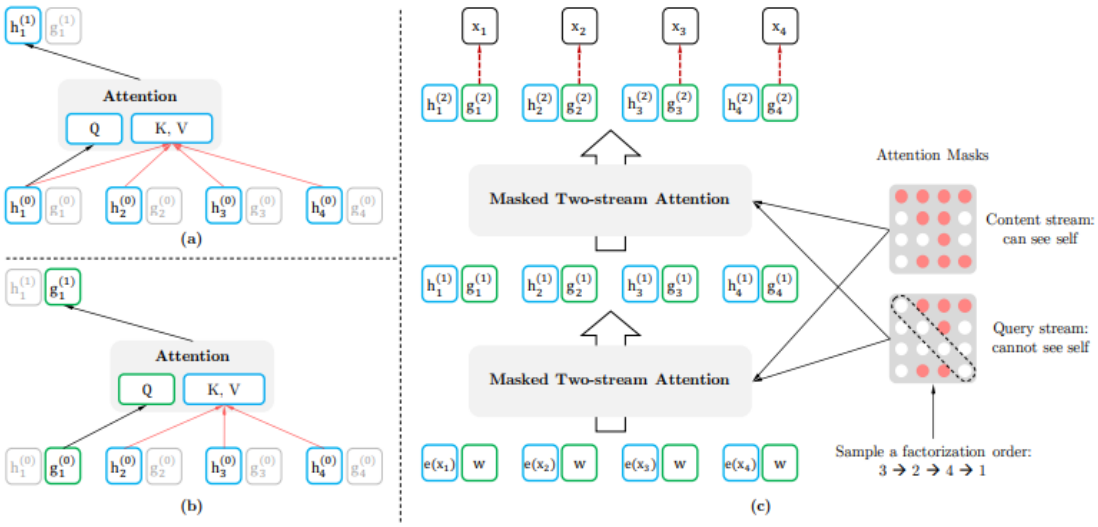

Figure 2.17: Given a permutation order, e.g. [3,2,4,1], XLNet generates the attention masks of the Content (a) and Query (b) streams in such a way that both can only attend to previous tokens in the factorisation order, and the query stream cannot attend to its own token content but only its position (c). Figure from [26].

To solve this problem, *XLNet* introduces ***Two-Stream Self-Attention*** for target-aware representations. As shown in Fig 2.17, now the model has two different self-attention mechanisms. The first one is ***"Content Stream"*** which works exactly as the *Transformer*'s self-attention. The second one is the ***"Query Stream"*** whose query Q attends to only the position of the token without looking at the content of the token. Therefore, each token x_i has now two hidden representations, one for content h_i and one for the position-awareness of the model g_i.

While during pretraining the Query stream is essential for the correct formulation of Permuted Language Modeling, at inference time, the architecture behaves similarly to a standard transformer, primarily utilizing the Content stream's self-attention.

MPNet and a unified pretraining strategy

In an attempt to take advantage of both the *autoregressive permuted strategy* developed by the *XLNet* strategy and the *Masked Language Modeling* of *BERT*, a novel architecture has been proposed. The intention was to unify these two approaches to pretraining and to develop one that could take advantage of both. The architecture presented was called ***MPNet: Masked Permuted Network*** [36].

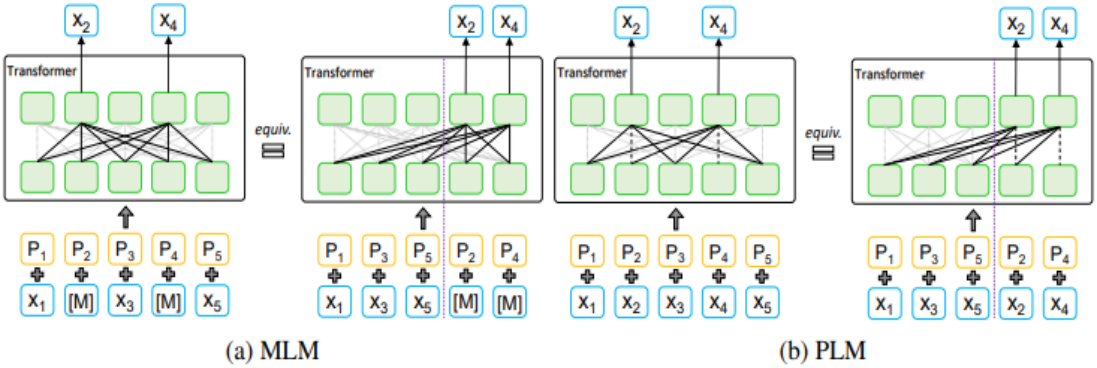

Figure 2.18: The unified view proposed in MPNet of MLM and PLM, where x_i represents the initial embedding of token i and P_i its position embedding. Figure from [36].

As shown in Fig 2.18, both the Masked Language Modeling and Permuted Language Modeling can be represented as permuting a sequence in way such that the tokens to predict are grouped on the rightest most part. Then MLM uses the positional information of all tokens and the content of the non masked ones to predict the masked tokens, therefore imposing Independence among them. Whereas PLM uses for each token only the information, both content and positional, of the previous tokens.

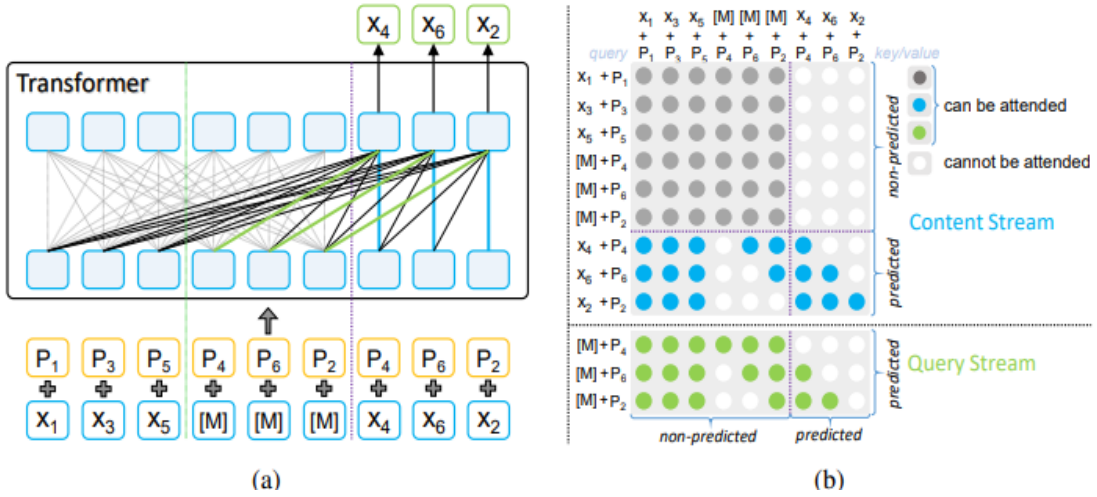

Figure 2.19: Attention mask system of MPNet. Reusing the idea of PLM of Content and Query attention stream. Figure from [36].

To model this unified objective, a new attention masking strategy is proposed by *MPNet* and visualized in Fig 2.19. Here the Content stream and the Query stream

are enriched by the representations of the masked tokens ant their positions within the sequence. Therefore, a token sequence like $[x_1, x_2, x_3, x_4, x_5, x_6]$ is enriched with its positional encoding $[P_1, P_2, P_3, P_4, P_5, P_6,]$ obtaining the initial position aware representation $x_i + P_i$ for each token. Then, the sequence is first permuted following a factorization order [1, 3, 5, 4, 6, 2], obtaining a pretraining sequence such as $[x_1 + P_1 +, x_3 + P_3, x_5 + P_5, x_4 + P_4, x_6 + P_6, x_2 + P_2]$ and then are added the mask special tokens *"[M]"* with the corresponding positional embeddings in the corresponding portion to predict. If the model has to predict the last 3 tokens in the permuted sequence (x_4, x_6, x_2), the masking addition will be $[M + P_4, M + P_6, M + P_2]$ and the input sequence to the model can be visualized as $[x_1 + P_1 +, x_3 + P_3, x_5 + P_5, M + P_4, M + P_6, M + P_2, x_4 + P_4, x_6 + P_6, x_2 + P_2]$. a visual representation is shown in Fig 2.20.

The last *MPNet* addition is modeling the Query stream by reusing the hidden states of Key and Values of the Content self-attention stream, while keeping generating a new Query value for the masked tokens to predict.

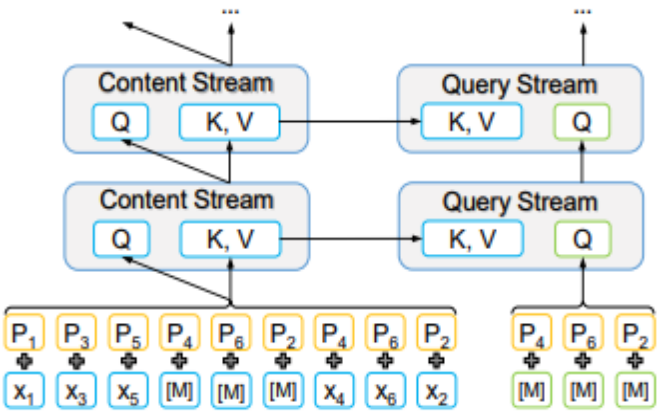

Figure 2.20: Attention system of MPNet. It reuse the hidden states from the Content stream to compute key and value in the Query stream. Figure from [36].

All these changes let the pretraining abandon the independence assumption over the masked tokens and enrich the context of the predicted tokens in the Permuted Language Modeling framework. *MPNet*, the adjustments proposed by *SentenceBERT* and the intuition about contrastive learning will be at the core of the methods developed for this thesis.

Chapter 3

Data Sources

This chapter introduces all the main data sources that were used to collect data for training and evaluating the whole tagging pipeline. First of all, section 3.1 explains the taxonomies used as reference to extract labels to tag the news stream, namely IPTC NewsCode (section 3.1.1) and ICD9 (section 3.1.2). These sections focus on why these taxonomy were specifically chosen, and suggest some previous work in which they have been used. Section 3.2 describes the datasets used in this thesis, with a mention of previous work and a detailed data analysis of them.

3.1 Taxonomies

A taxonomy is defined as a hierarchical classification scheme generally used to index and organize knowledge. It can be thought of as a division of the knowledge space into non-overlapping areas, the specificity or granularity of which depends on the level of hierarchy at which one wishes to remain. So one can think of the taxonomy as a tree structure where the root is the area of knowledge it relates to, and the deeper one moves through the taxonomy, the more specific the characterization.

The strategy adopted in this work is to use established taxonomies in different domains to extract reliable and meaningful labels with which to tag the text data. Note that this choice is largely supported by the literature [37] [38] [39], where the ICD9 taxonomy is often used as a reference for categorising medical records.

3.1.1 IPTC

The first taxonomy presented relates to the field of knowledge of news topics. IPTC stands for International Press Telecommunications Council and the IPTC taxonomy called NewsCodes MediaTopic[1] is a hierarchical scheme that categorizes the topics a news item can report on. The first level of this taxonomy, as it is possible to see in Figure 3.1, corresponds to the courser division of the knowledge space and consists of 17 different concepts. that vary from *"arts, culture, entertainment and media"* to *"weather"*. Among these topics, the main goal the first level want to achieve is to be able to recognize the topic *"health"*. The second level of the MediaTopic *"health"* contains the following categories:

- *"disease and condition"*
- *"health facility"*
- *"health organisation"*
- *"government health care"*
- *"health insurance"*
- *"private health care"*
- *"medical profession"*
- *"non-human diseases"*
- *"public health"*

[1] https://show.newscodes.org/index.html?newscodes=subj&lang=en-GB&startTo=Show

- *"health treatment and procedure"*

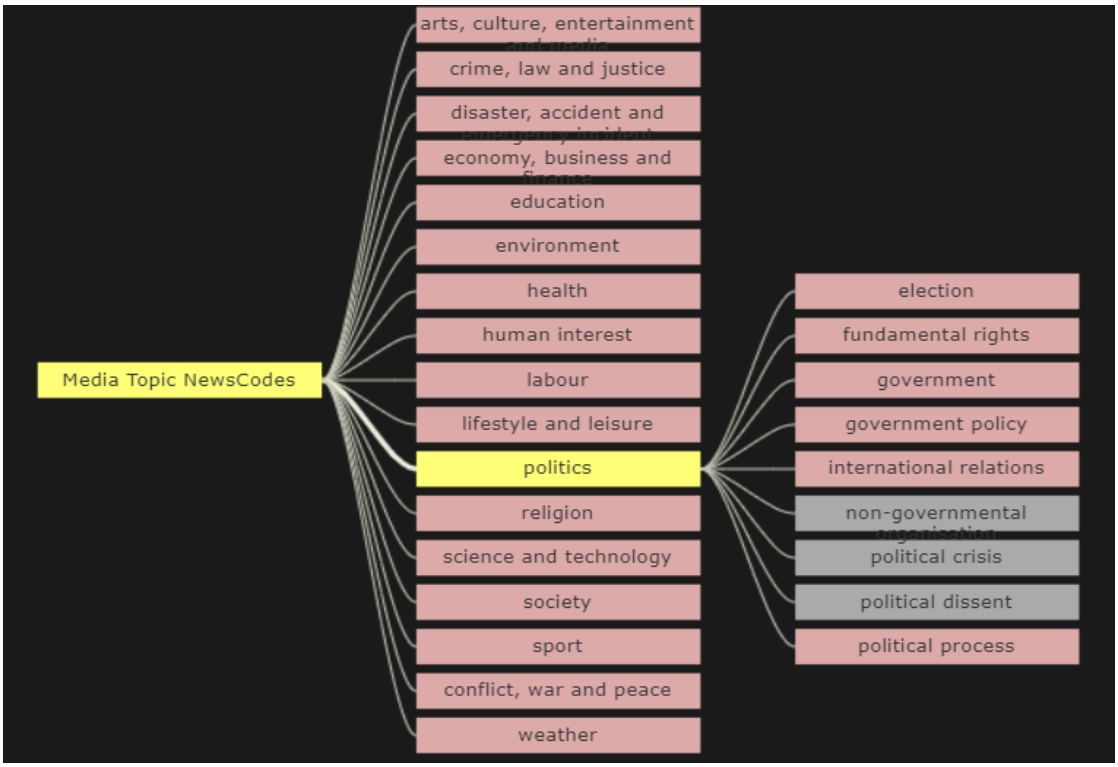

Figure 3.1: IPTC taxonomy of Media Codes, all first levels and second level of media topic politics. The color pink indicates that the node can be deeper explored.

It is possible to see both in Figure 3.1 and in the above example that, semantically, a good way to define each first level element of the taxonomy is to look at what elements make up its own deeper level. This consideration will be crucial in the definition of the labels for the tagging system.

3.1.2 ICD9

The second taxonomy used in this work is the ICD9 codes taxonomy [2]. The acronym ICD-9-CM stands for International Classification of Diseases, 9th revision - Clinical Modification. It is an international classification system for diseases, injuries, surgeries, and diagnostic and therapeutic procedures proposed by the World Health Organization. ICD-9-CM coding aims to provide a standardized and unambiguous nomenclature for health conditions to facilitate the collection and analysis of health data at international levels. Through standardization, it also enhance the monitoring of epidemiological trends in diseases, quality and adequacy of health care, resource use and health care effective financing.

The ICD9 codes are numerical or alpha-numerical codes relative to specific diseases and the hierarchical structure helps in grouping them together.

- **Numerical Codes**: these are codes that consist of at least 3 digits and describe the disease (e.g. 480 is *"Viral Pneumonia"*), which can be further specified with decimal numbers (e.g. 480.3 is *"Pneumonia due to SARS associated coronavirus"*). These codes follow the following grouping system:
  - **(001-139)** *Infectious And Parasitic Diseases*
  - **(140-239)** *Neoplasms*
  - **(240-279)** *Endocrine, Nutritional And Metabolic Diseases, And Immunity Disorders*
  - **(280-289)** *Diseases Of The Blood And Blood-Forming Organs*
  - **(290-319)** *Mental Disorders*
  - **(320-389)** *Diseases Of The Nervous System And Sense Organs*
  - **(390-459)** *Diseases Of The Circulatory System*
  - **(460-519)** *Diseases Of The Respiratory System*
  - **(520-579)** *Diseases Of The Digestive System*
  - **(580-629)** *Diseases Of The Genitourinary System*
  - **(630-679)** *Complications Of Pregnancy, Childbirth, And The Puerperium*
  - **(680-709)** *Diseases Of The Skin And Subcutaneous Tissue*
  - **(710-739)** *Diseases Of The Musculoskeletal System And Connective Tissue*
  - **(740-759)** *Congenital Anomalies*

[2] https://ftp.cdc.gov/pub/health_Statistics/nchs/Publications/ICD9-CM/2007/

  - **(760-779)** *Certain Conditions Originating In The Perinatal Period*
  - **(780-799)** *Symptoms, Signs, And Ill-Defined Conditions*
  - **(800-999)** *Injury And Poisoning*

- **V Codes (V01-V91)**: these are alphanumerical codes starting with letter *V* that are used to identify *Supplementary Classification Of Factors Influencing Health Status And Contact With Health Services*.

- **E Codes (E000-E999)**: these are alphanumerical codes starting with letter *E* that are used to identify*Supplementary Classification Of External Causes Of Injury And Poisoning*.

3.2 Datasets

Several datasets have been used in this thesis, and some have been specifically created to address specific problems, such as the lack of public and tagged news datasets with a specific focus on health-related news, or the quality of the textual data in the available datasets. The section 3.2.1 introduces the World Health Organization dataset, which is generated through an openly available API. The section 3.2.2 instead presents a BBC dataset that is mainly used for topic classification of news. How these datasets are manipulated and processed for the scope of this thesis is instead presented in chapter 4. In the last two subsections two dataset are introduced: MIMIC-III (section 3.2.3) and GDELT (section 3.2.4).

3.2.1 World Health Organization

World Health Organizaton[3] is a specialized United Nations institute for health. Among its several interestes, for several years it explored how digitization and technology can help global health. Some examples are the ICD9 coding, introduced in subsection 3.1.2, but also through the distribution of updated datasets and publication of freely accessible online news.

Their news site is a globally recognized source and is well structured and features several tags of interest. These include *"disease outbreaks"*. This is precisely the type of news that TrustAlert wants to intercept in *GDELT* stream which is analysing. By using the API[4] provided by WHO, it was possible to collect all the news related to *"disease outbreaks"* and capture 2805 different news from 01.01.2020 to 31.12.2023 (note that the time frame was chosen to be compatible with the GDELT dataset available at LINKS, but the dataset thus obtained can still be extended)

[3]https://www.who.int/

[4]https://www.who.int/api/news

3.2.2 BBC - British Broadcasting Corporation

BBC or British Broadcasting Corporation is a recognized and respected news source around the world. An openly accessible dataset[5] regularly collects its news and categorizes it into five different topics:

- *Business*
- *Entertainment*
- *Politics*
- *Sport*
- *Tech*

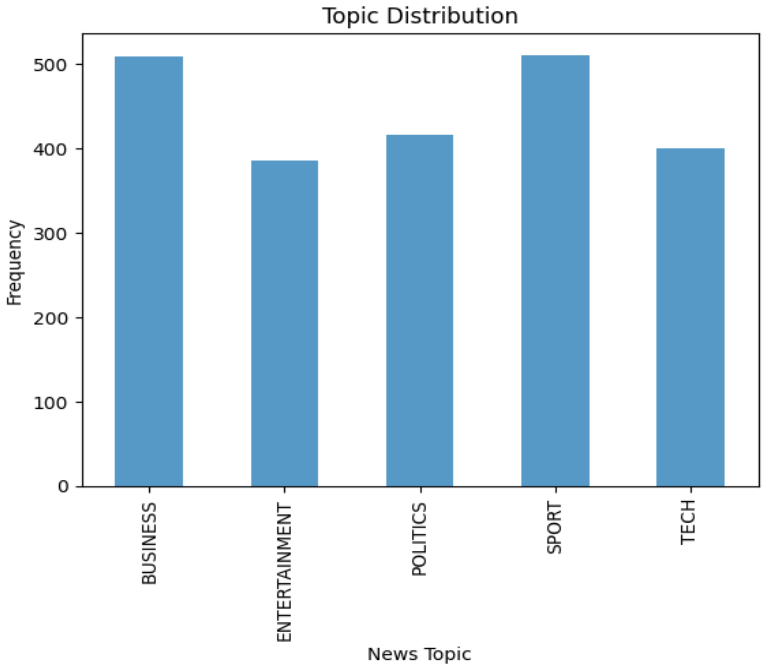

Figure 3.2: Topic Distribution in BBC dataset.

As we can see in Figure 3.2, this dataset is almost balanced with at least 386 news per topic and a total of 2225 news. However, it lacks the main category on which this project would like to focus, namely health-related news.

[5]https://www.kaggle.com/datasets/gpreda/bbc-news

3.2.3 MIMIC-III

MIMIC-III is a database of anonymized health-related data from patients treated in the intensive care unit of Beth Israel Deaconess Medical Center between 2001 and 2012. The database contains information of various data types, such as categorical data on demographics, time series derived from electronic measurements of bedside vital signs, and many other information like lab test results, procedures, medications, nursing staff records, imaging reports, and mortality.

MIMIC supports a variety of analytic studies, from epidemiology to clinical decision rule improvement to electronic tool development.

During this work MIMIC was investigated to look at the feasibility of its implementation as a test suite for the tagger on the ICD9 coding task.

Given the nature of this work, the type of data analyzed is the textual one. In the database there is table of notes relative to each hospitalization called *NOTEEVENTS*, the types of possible textual data and their distribution in MIMIC-III is reported in Figure 3.3.

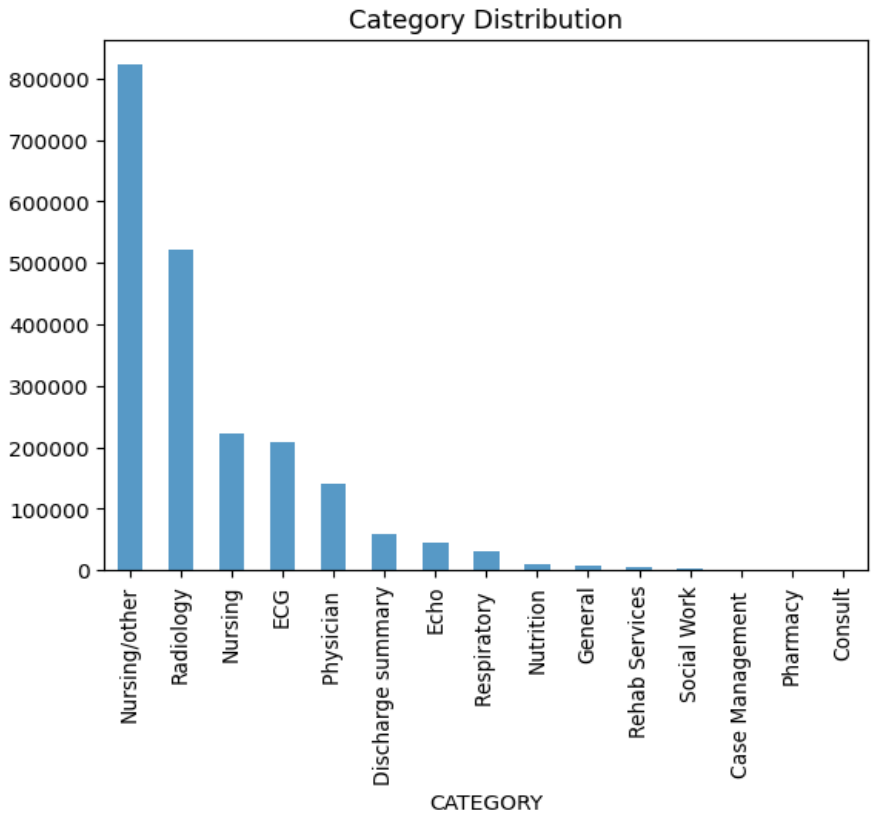

Figure 3.3: Category Distribution of textual data in MIMIC-III dataset.

Among these variety of textual data there are the *"Discharge summary"* that includes medical history, diagnostic results, surgical procedures, discharge instructions, etc. Each admission record is assigned a set of the most important ICD-9 codes by the experts of the ICU of Beth Israel Deaconess Medical.

By grouping the ICD9 codes assignment by admission code of each hospitalization, it is possible to have a dataset that relates the discharge notes composed by textual data with the corresponding ICD9 codes. In doing so, it is possible to obtain a dataset of 59652 different discharge summaries with their codes.

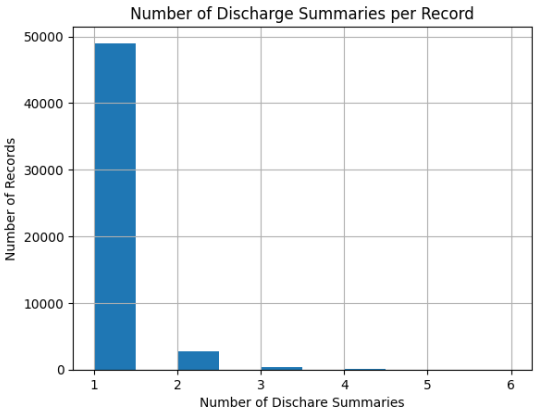

Figure 3.4: Number of discharge summary distribution

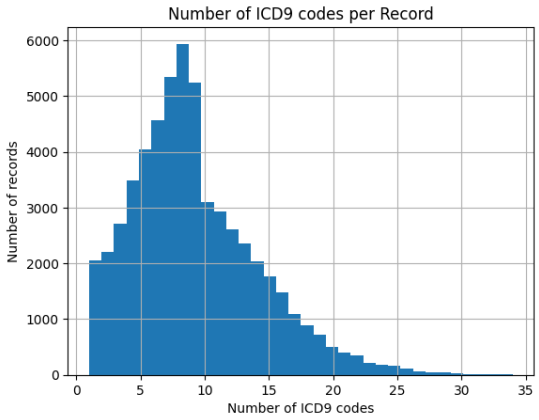

Figure 3.5: Number of ICD9 codes distribution

As we can see in Figure 3.4 and in Figure 3.5 the vast majority of the records has one only discharge notes while the number of ICD9 codes assigned to each of them vary a lot. This can cause some problem both in the definition of the multi-label classification problem and in the one of the tag retrieval defined in section 4.1.

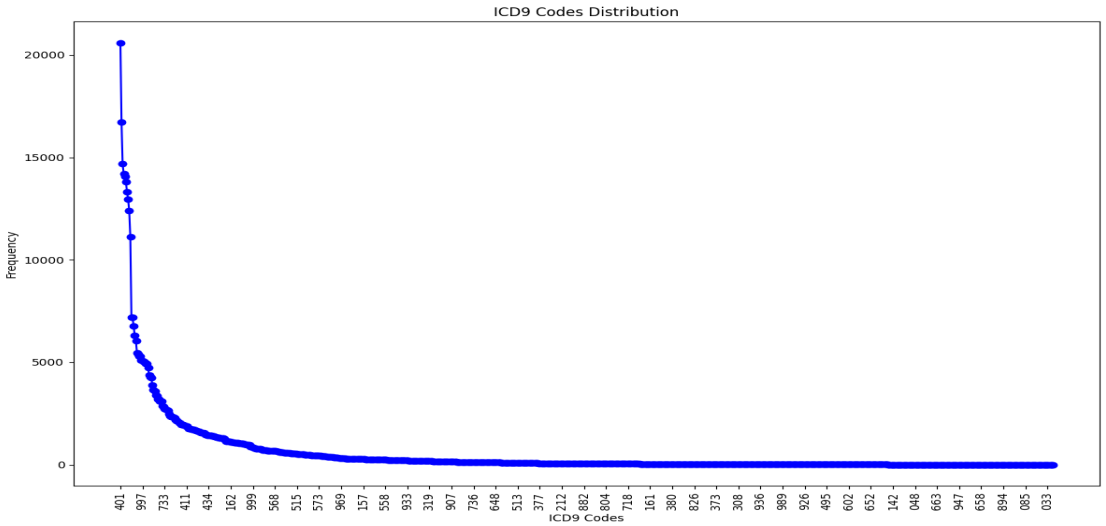

Figure 3.6: ICD9 codes Distribution in MIMIC-III dataset.

As we can see in Figure 3.6, another major problem is the distribution of the ICD9 codes themselves. Indeed, it seems to be a long-tailed distribution, with many examples of a minority of codes and some of them with few or none.

To have a quantitative example, the first five codes sorted by frequency with their relative occurrence are reported below:

- **(401)** *Essential hypertension*: 20592 occurrences
- **(427)** *Cardiac dysrhythmias*: 16731 occurrences
- **(276)** *Disorders of fluid, electrolyte, and acid-base balance*: 14682 occurrences
- **(272)** *Disorders of lipoid metabolism*: 14185 occurrences
- **(414)** *Other forms of chronic ischemic heart disease*: 14054 occurrences

Another important piece of information to understand how this dataset is biassed is the fact that the ICD9 taxonomy used as tags includes 922 different codes and 27 of these codes appear only once in the MIMIC III dataset, while 80 never appear (e.g. **001** *Cholera*).

To avoid the problem related to the label imbalance and missing codes some previous works done on MIMIC III such as [37] and [38] suggest to restrain the problem to the filter the dataset keeping only the most frequent ICD9 codes. Common choices are keeping only thr top 10 and top 50 ICD9 codes.

The resulting dataset will be referred to as *"MIMIC III TOP 10"* or *"MIMIC III TOP 50"* and it is composed by 44304 different discharge notes with their relative ICD9 codes that appear into the top 10 or top 50 most frequent ICD9 codes considering MIMIC III ICD9 codes distribution reported in Figure 3.6.

3.2.4 GDELT

GDELT[6] or Global Database of Events, Language, and Tone is a database created by Google that collects news from around the world. It monitors the world's news media in print, broadcast and web formats and translates every news item from over 100 monitored languages into English. Using Google own words: *"the GDELT Project is a real-time open data global graph over human society as seen through the eyes of the world's news media, reaching deeply into local events, reaction, discourse, and emotions of the most remote corners of the world in near real-time and making all of this available as an open database to enable research over human society"*.

GDELT is the news stream that the TrustAlert Project aims to monitor to gather news on health issues. In addition, the nature of GDELT allows the stream to be split by geographic areas of interest and the system to be used for a specific target, such as a city, a country or an entire continent. To get an idea of the amount of data collected by GDELT, it suffices to point out that the system stores more than 250 million events recorded worldwide since 1979, allowing anyone to access them and extract information.

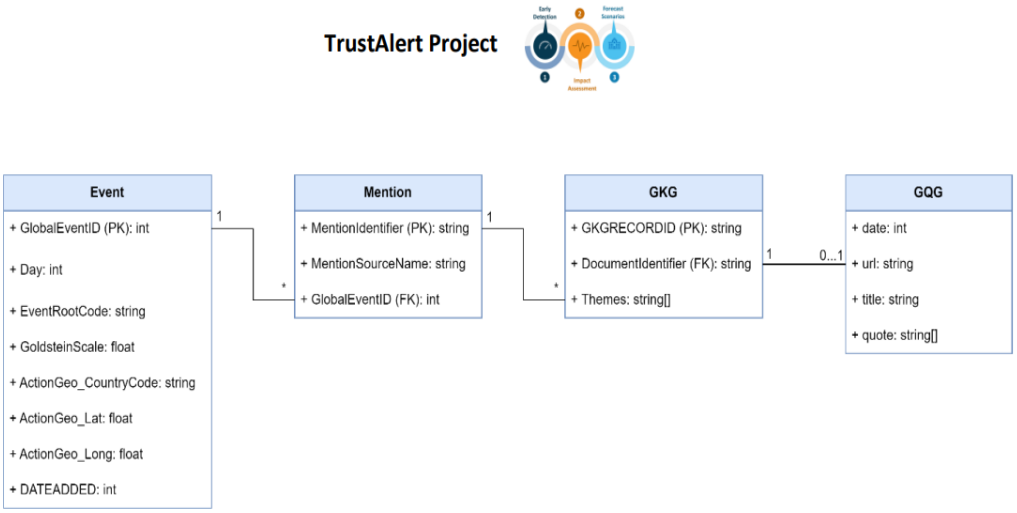

Figure 3.7: GDELT main tables and their relationships

GDELT is structured like a real-time event collector and has various tables. Of the entire database, LINKS Foundation has decided to focus on the tables: *Event*, *Mentions*, *GKG (Global Knowledge Graph)* and *GQG (Global Quotations Graph)*. The relationships between these tables are shown in the Entity Relationship (ER)

[6]https://www.gdeltproject.org/#watching

schema, which can be seen in Figure 3.7. These tables are published as archive files, each containing a CSV serialization of the corresponding table. The table archives are created every 15 minutes and enable knowledge extraction almost in real time. A brief description of the main tables is reported below:

- *Event*: this table collects various events and assign to them a unique identifier.
- *Mentions*: this table reports on all the mentions across media collected that refers to a specific event tracked by *Event* table.
- *GKG (Global Knowledge Graph)*: this table provides expanded context by capturing people, organizations, companies, unique places, millions of topics and thousands of emotions from each article.
- *GQG (Global Quotations Graph)*: this table contains text excerpts from articles. Each line of the data set corresponds to a specific article and contains its quotations. Unlike the other tables, The *GQG* dataset is updated every minute.

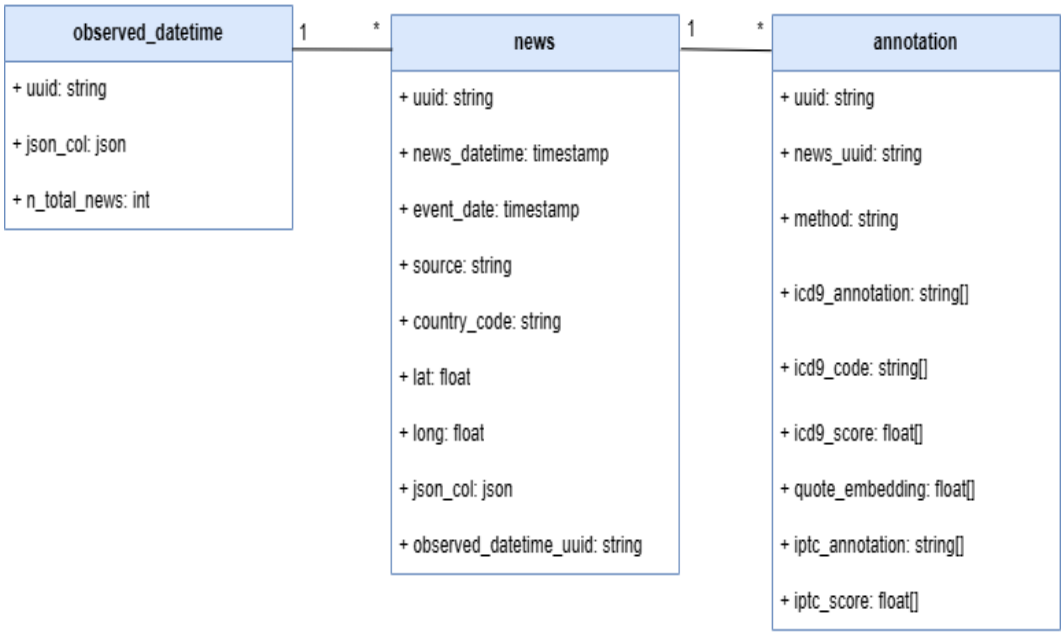

Figure 3.8: Data model adopted by LINKS Foundation.

The GDELT dataset was queried through SQL by LINKS Foundation in order to make a smaller and narrower version available locally focused on data since 2018. A different data schema with three main tables was chosen for the local version, which can be seen in Figure 3.8. The main tables of this version of GDELT are:

- *Observed Datetime*: this table tracks the data downloaded from GDELT using their timestamp as unique identifier.
- *News*: this table contains metadata about the news collecte by GDELT.
- *Annotation*: this table structures the information produced from the semantic analysis and filtering of the GDELT news.

Chapter 4

Method

This chapter introduces all the important elements that contribute to the functionality of the entire tagging pipeline. First of all, section 4.1 formally defines the task that is set for the tagger system. Section 4.2 describes how the taxonomies introduced in 3.1 are manipulated to extract labels with the maximum amount of information from them. Section 4.3 describes the datasets created specifically for this work using data sources introduced in 3. Finally, section 4.4 presents the model architectures and their training strategies.

Before we get into the technical details, it is important to reiterate the constraints of the TrustAlert project, as it has some serious implications for technical decisions. Namely, the main goal is a disease outbreak alert system that helps hospitals and healthcare facilities track anomalies in these types of trends so as not to be caught unprepared. This means that the evaluation suite should focus more on ICD9 codes of communicable diseases rather than the more generic and common codes. Another limitation is the style of the text data. Since they are texts from global news sources, they are neither very technical nor use too specific terms. These two considerations made it clear that the available dataset such as MIMIC-III, which was presented and analyzed in section 3.2.3, is insufficient, so we had to look for another dataset to work with. These considerations resulted in the creation of a custom dataset presented in section 4.3.1.

Another consideration is that while this work is based on a specific taxonomy, namely the ICD9 taxonomy presented in section 3.1.2, the landscape of science-related taxonomies is frequently changing. There is currently an ICD10 version of the ICD taxonomy and the World Health Organization is already developing the ICD11 taxonomy. This situation highlights the importance of having a model that is flexible in terms of the tags we want to use. The chosen approach is therefore to explore techniques, such as *pre training* techniques and *contrastive learning*, that are not tied to the number or type of these tags. In other words, the lack of

labeled datasets and the flexibility required to focusing on the characteristics of the latent spaces generated by these models and the data they works with, rather than focusing on the specialization of a classifier on this specific task. In fact, the chosen approach is to use a backbone pre-trained model to encode both news and labels in its latent space and use a distance metric to retrieve the nearest tag to the news. This choice let the model be fine tuned through contrastive learning techniques while keeping its flexibility in being used as a zero-shot model on unseen tags.

To sum up the main constraints:

- *Lack of labelled dataset.*
- *Available datasets not aligned with the real case scenario of TrustAlert.*
- *Necessity not to be constrained by the tag set.*

To sum up the choices taken:

- *Creation of a dataset that resembles the context TrustAlert will be deployed.*
- *Focus on zero-shot capality of the models inspected.*
- *Training in contrastive learning settings.*

4.1 Task Formulation

The core of this work can be represented as a two-stage tagging problem. Once a message source is selected, such as GDELT introduced in section 3.2.4, the *"phase one"* tagger must be able to distinguish between news reporting on *health* and all others. This behaviour is formalized in section 4.1.1.

While the *"phase one"* tagger limits the message stream to health-related news, the *"phase two"* tagger is responsible for tagging each of these news with the corresponding disease using the ICD9 taxonomy, presented in section 3.1.2, as a reference. This specific behaviour is introduced in section 4.1.2.

This two-stage tagging pipeline generates disease specific time series, and graphs of these time series are displayed on a visualization dashboard developed by LINKS on Grafana.

4.1.1 Phase One Tagger Task

The tagger of the *"phase one"* is responsible for the correct assignment of the tag *"health"* to the news stream. Specifically, the tags available for each news are the list of codes in the IPTC taxonomy, which is presented in detail in section 3.1.1.

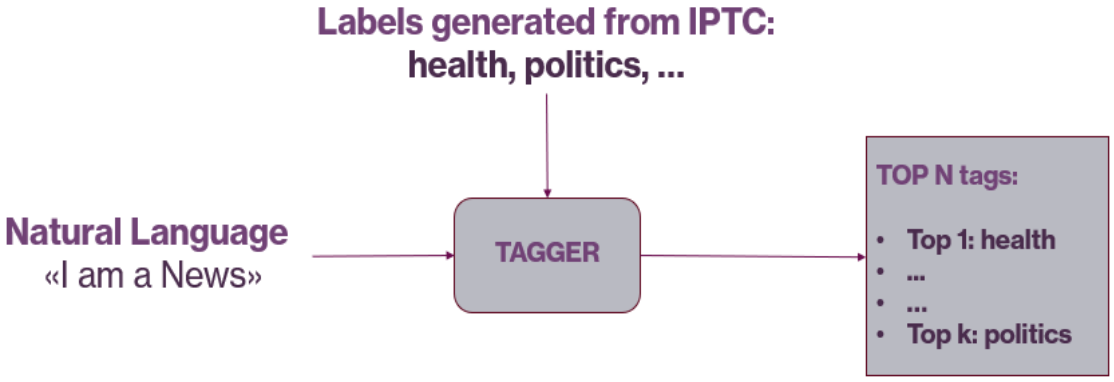

Figure 4.1: Visualization of the *"phase one"* tagging task. All the news that will have *"health"* as Top1 tag will be kept for the second level of tagging.

The expected behaviour of this pipeline (Fig 4.1) can be summarized as follows:

- Let x be a news in plain text format.
- Let $tagset_{IPTC}$ be the set of all possible tags generated from IPTC taxonomy.

$$tagset_{IPTC} := \{tag_{IPTC_1}, tag_{IPTC_2}, ..., tag_{IPTC_N}\}.$$

- Let $Tagger(x\,,\,tagset)$ be the system with a backbone model able to process the text x and retrieving one $tag \in tagset$ or a ranked subset of top N tags $\{tag_i\} \in tagset$.
- Let $sim_{Tagger}(x, tag)$ a similarity metric to compare the news with respect to all the labels available in $tagset_{IPTC}$ using the representation of x and tag obtained through the $Tagger$.

$$Tagger(x\,,\,tagset_{IPTC}) = \arg\max_{i \in tagset_{IPTC}} \{\; sim_{Tagger}(x, tag_{IPTC_i}) \;\}$$

$$if \quad Tagger(x\,,\,tagset_{IPTC}) == \text{"health"}$$
$$then \text{ keep } x$$

In simple words, the tagger retrieves the tag_{IPTC} that is most similar to the news based on a similarity metric search performed between the news x and all the tags preset in the $tagset_{IPTC}$ using a specific similarity sim.

Then, based on the tagging results it is decided whether to keep the news or not. The ideal output of this first level of tagging is a perfectly divided stream in *"health"* and *"non-health"* news.

4.1.2 Phase Two Tagger Task

In relation to the task of the *"phase one"*, the *"second phase"* now focuses on news reporting on health topics. The specific task of the *"second phase"* tagger is to assign a disease-specific code to each news item that is filtered by the first level tagger.

The disease codes are generated from the ICD9 taxonomy presented in 3.1.2.

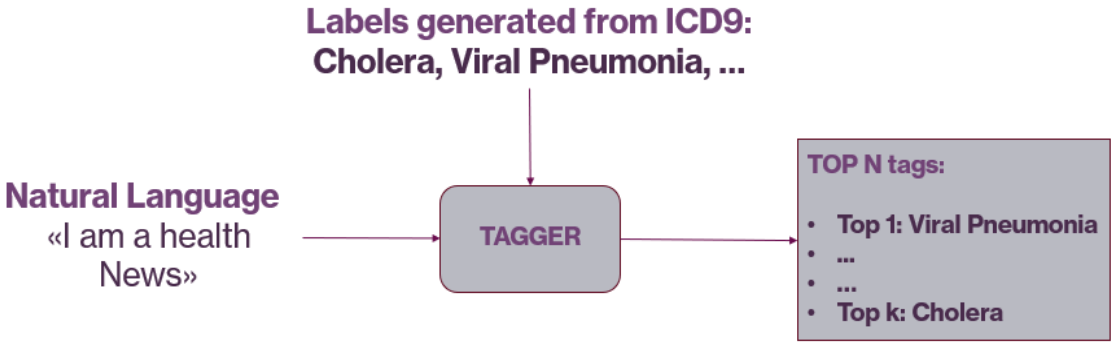

Figure 4.2: Visualization of the *"phase two"* tagging task. All the news will be tagged with their Top1 tag.

The expected behaviour of this second pipeline (Fig 4.2) can be summarized as follows:

- Let x be a news in plain text format filtered by the first level tagger.
- Let $tagset_{ICD9}$ be the set of all tags generated from ICD9 taxonomy.

$$tagset_{ICD9} := \{tag_{ICD9_1}, tag_{ICD9_2}, ..., tag_{ICD9_N}\}.$$

- Let $Tagger(x\,,\, tagset)$ be the system with a backbone model, possibly different from the *"phase one"* tagger, able to process the text x and retrieving one $tag \in tagset$ or a ranked subset of top N tags $\{tag_i\} \in tagset$.
- Let $sim_{Tagger}(x, tag)$ a similarity metric to compare the news with respect to all the labels available in $tagset_{ICD9}$ using the representation of x and tag obtained through the $Tagger$.

The *"phase two"* tagger tagging functionality can be described as:

$$Tagger(x\,,\, tagset_{ICD9}) = \arg\max_{i \in tagset_{ICD9}} \{\; sim_{Tagger}(x, tag_{ICD9_i}) \;\}$$

Therefore, the tagger finds the tag_{ICD9} that is most similar to the news based on a metric similarity search between the news x and all tags preset in the $tagset_{ICD9}$ using a certain similarity sim.

The ideal output of this second level of tagging is a perfectly tagged stream with each news tagged with the diseases it reports on.

4.2 Taxonomy Expansion

As described in section 4.1, the tagger uses a tag set to select the tag to be assigned to each news. This tag set is generated based on the taxonomies presented in section 3.1 using various techniques presented in this section.

The basic idea is that the similarity used by the tagger is intended to be a semantic similarity, which is why we want each tag to be as meaningful as possible in terms of semantic content. While the news consist of a considerable amount of text, as we can see in the description of the datasets in section 3.2, the taxonomy terms are quite short, e.g. the tag we want to assign to the health-related news is simply *"health"*.

Even if these terms are semantically correct, it is possible that not enough semantic information can be extracted from a single word. For this reason, some techniques for semantically extending the taxonomy are being explored.

4.2.1 Primary level taxonomy expansion

This technique is the simplest way to extract a tag from a taxonomy. Looking at the taxonomy as a nested dictionary of concepts, each primary key becomes a tag in the tag set. Therefore, each top level concept becomes a tag that can be chosen by the tagger.

As expected, these tags are the shortest ones and the one less specific, corresponding to top level broader concepts.

Using as reference the ICD9 taxonomy, presented in section 3.1.2, the tag related to *Cholera* is simply the first level entry of the taxonomy *"Cholera"*.

- **Primary**: *"Cholera"*

4.2.2 Secondary level taxonomy expansion

This technique builds on the idea of the first. Taking advantage of the fact that every taxonomy we use has at least one depth level below the top level, it is possible to extract and combine both the primary and secondary level information for each concept to obtain more meaningful tags.

Taking again the ICD9 taxonomy, presented in section 3.1.2, as reference and specifically the code **(001)** *"Cholera"* we can see how the code is expanded:

- **Primary**: *"Cholera"*
- **Secondary**: *"Cholera: Due to Vibrio cholerae, Due to Vibrio cholerae el tor, Cholera, unspecified "*

4.2.3 Description taxonomy expansion

This technique specifically refers to the ICD9 taxonomy. As it is possible to see, the previous technique proposes an expansion that is still very specific and does not add very much as context. Therefore, this technique proposes an approach of definition retrieval for each code.

Given a code, we look online for the definition of the diseases through a site that collects ICD9 code definitions (www.icd9data.com) and concatenate it with the code name. Unfortunately, not all the codes have an available definition, therefore for the missing ones we re-use the technique of concatenating the second level of the taxonomy.

Taking two examples with and without a definition we can see that when a definition is not available the last two techniques behave the same way.

With a definition, code **(001)** *"Cholera"*:

- **Primary**: *"Cholera"*
- **Secondary**: *"Cholera: Due to Vibrio cholerae, Due to Vibrio cholerae el tor, Cholera, unspecified"*
- **Definition**: *"Cholera: Acute diarrheal disease endemic in India and southeast Asia whose causative agent is vibrio cholerae can lead to severe dehydration in a matter of hours unless quickly treated."*

Without a definition, code **(274)** *"Gout"*:

- **Primary**: *"Gout"*
- **Secondary**: *"Gout: Gouty arthropathy, Gouty nephropathy, Gout with other specified manifestations, Gout, unspecified"*
- **Definition**: *"Gout: Gouty arthropathy, Gouty nephropathy, Gout with other specified manifestations, Gout, unspecified"*

4.3 Datasets Generation

During the development of the TrustAlert project, several datasets were used and some others were specifically created to solve certain problems, such as the lack of public and labelled news datasets with a specific focus on health-related news or the quality of textual data in the available datasets. In subsection 4.3.1 is possible to find how the news dataset was created, while in subsection 4.3.2 it is possible to see how this dataset was turned into a dataset to evaluate ICD9 coding capabilities of the tagging system.

4.3.1 Developed news dataset

As specified in subsection 4.1.1, the first capability of the tagger system to be tested is whether it is able to distinguish health-related news from the others. After a careful examination of the openly available datasets, it was clear that, as of now, there were none that simultaneously fulfilled the following mandatory characteristics:

- Text format and style resembling the one of online news stream. This means a fluent and freely formulated text without overly specific terms from the medical field.
- Labels (or more in general features) that let distinguish between news reporting on health topic and the other ones.

The first constraint ensures that the test suite resembles the real scenario in which the tagging system is used. The second is imperative to have a ground truth against which the results of the system can be compared. Hence, the idea of combining two reliable news sources such as the World Health Organization (section 3.2.1) and the BBC (section 3.2.2) to produce a labelled and reproducible dataset.

Final Dataset

To create a final dataset able to respect both the constraint introduced in subsection 4.3.1, the idea was to fuse the World Health Organization dataset giving to those the ground truth label *"health"* while keeping the news of BBC with their original labels or giving to them the ground truth label *"not health"*. Remember that, as we can see on Figure 3.2, the BBC news surely are not reporting on health topics.

As we can see in Figure 4.3, the original datasets has a news length distribution that is comparable. Instead, the final label distribution, considering a binary classification task of discerning *"health"* news from *"non-health"* news can be seen in Figure 4.4. The final length of the dataset is of 5030 different news.

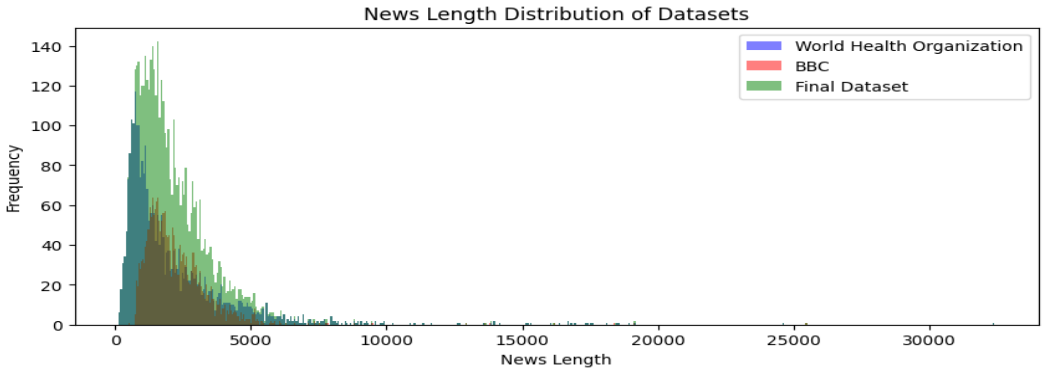

Figure 4.3: Topic Distribution in all news datasets.

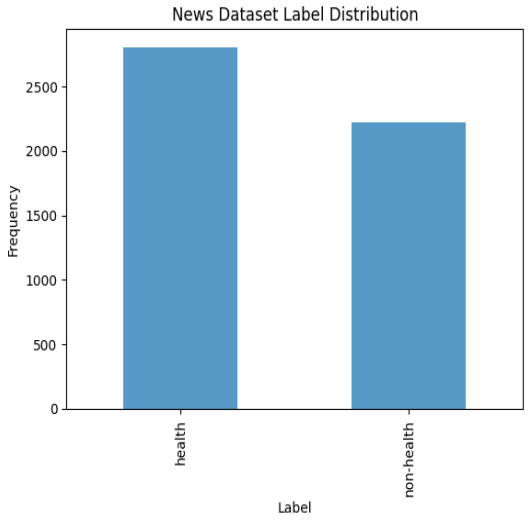

Figure 4.4: Label Distribution in the generated news dataset.

The dataset generated as explained will be used to evaluate, as explained in section 4.1, the tagger system ability to distinguish among the news that report on health topic such as diseases, as reported in the World Health Organization news on diseases outbreaks, and the other news that will be considered irrelevant for the TrustAlert project.

4.3.2 Developed dataset adapted to ICD9: "WHO-ICD9"

The news dataset generated as in subsection 4.3.1 is useful to quantify the ability of the developed tagger system to recognize health-related messages. However, it is also interesting to investigate a more specific capability of the tagger system: its ability to understand which disease the news is reporting on.

To evaluate this specific capability the only subset of news that can be useful is clearly the one retrieved through the World Health Organization API, as explained in subsection 3.2.1.

After identifying the subgroup of interest, the challenge was to transform this unlabeled corpus into a labelled dataset. The simplest but most effective idea was to use the title of the news and their text to search for a perfect match among the textual description of the ICD9 codes.

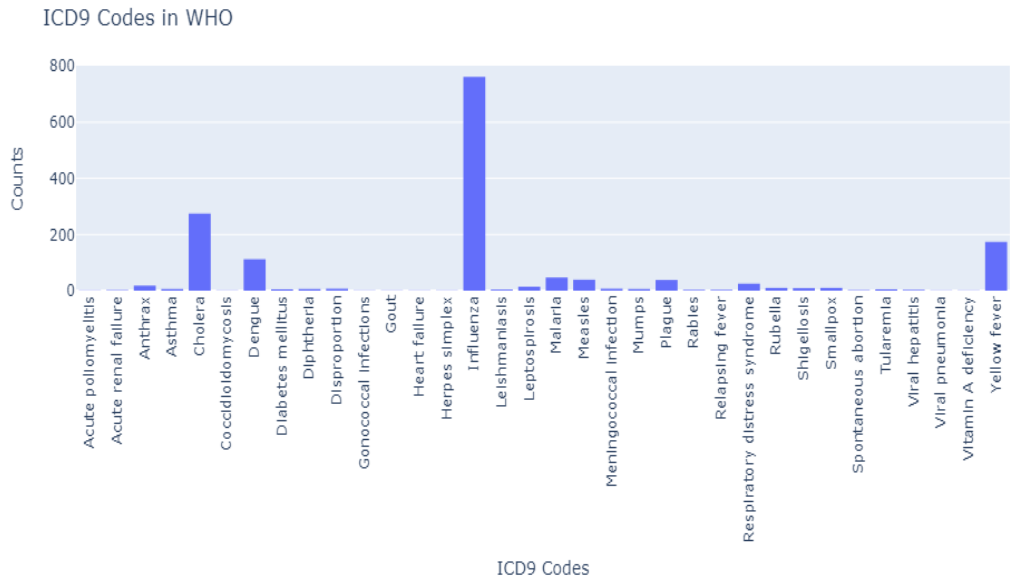

Figure 4.5: ICD9 codes distribution in the generated news dataset with *perfect matching strategy.*

The perfect matching strategy gave us a dataset that follows the ICD9 code distribution shown in Figure 4.5. Although this strategy is promising, the number

of ICD9 codes found per news in Figure 4.6 shows that a large proportion of the news in the WHO dataset would be lost if this simple strategy were implemented. Specifically, only 1477 of the 2805 original news were assigned a golden truth and looking at the ICD9 code distribution in Figure 4.5 it is possible to see that few codes have a significant amount of examples, such as *"influenza"*.

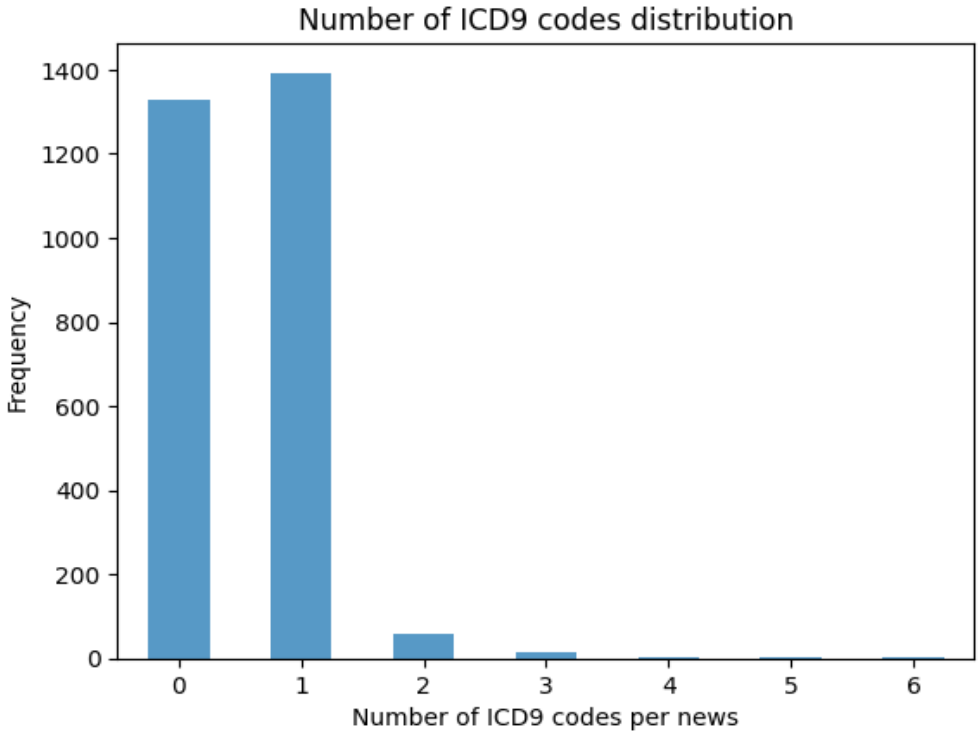

Figure 4.6: Number of ICD9 codes in the generated news dataset with *perfect matching* strategy.

In reviewing the news of WHO news, which did not match perfectly, it was found that some diseases were misspelled and some other acronyms were used, so a series of manual rules were applied to correctly label these news. The rules applied are described below:

- If a perfect match is found no further examination is done.
- if *"Meningococcal"* is found in the news
  then **(036)**:*"Meningococcal infection"* is applied.
- if *"Polio"* is found in the news
  then **(045)**:*"Acute poliomyelitis"* is applied.

- if *"HIV"* is found in the news
  then **(042)** *"Human immunodeficiency virus [HIV] disease"* is applied.

- if *"Ebola"* or *"hemoraagic fever"* is found in the news
  then **(065)**: *"Arthropod-borne hemorrhagic fever"* is applied.

- if *"Coronavirus"* or *"SARS"* is found in the news
  then **(480)**:*"Viral Pneumonia"* is applied.

From now on the dataset prepared in this way will be referred to as *"perfect matching plus"* to make it clear which version of the dataset the experiments and statements refer to.

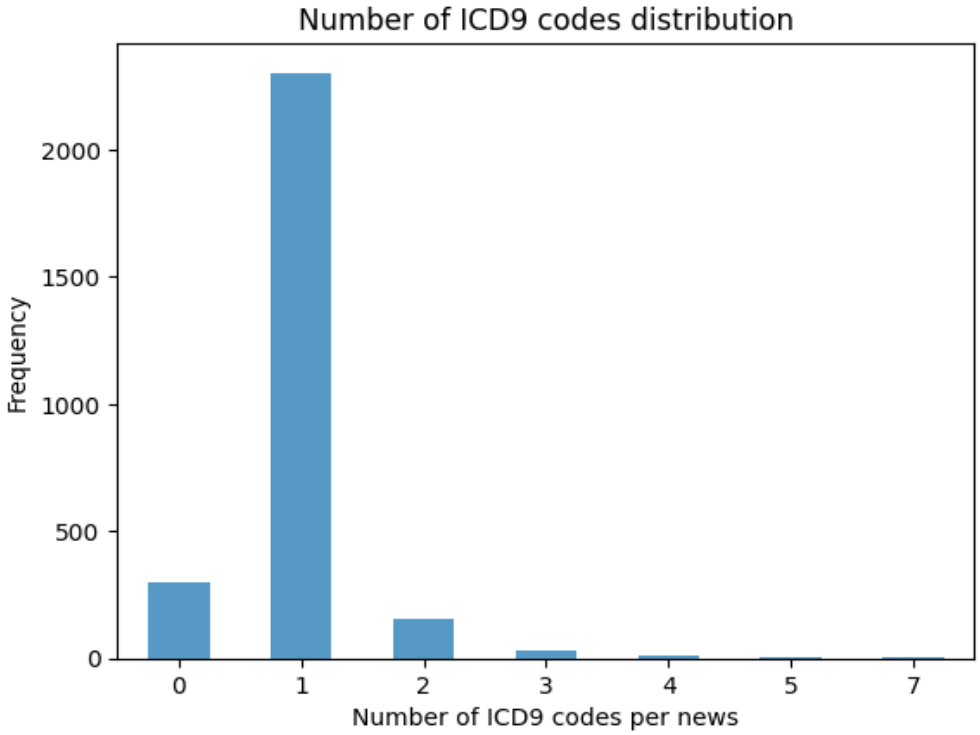

Figure 4.7: Number of ICD9 codes in the generated news dataset with *perfect matching plus* strategy.

Looking at Figure 4.7 where is reported the final number of ICD9 assigned per news in it is possible to see that using the *perfect matching plus* strategy to assign golden truth codes it was possible to cover a significant part of the original WHO dataset. Specifically, 2504 of the 2805 original news were assigned a golden truth and the ICD9 codes distribution is reported in Figure 4.8.

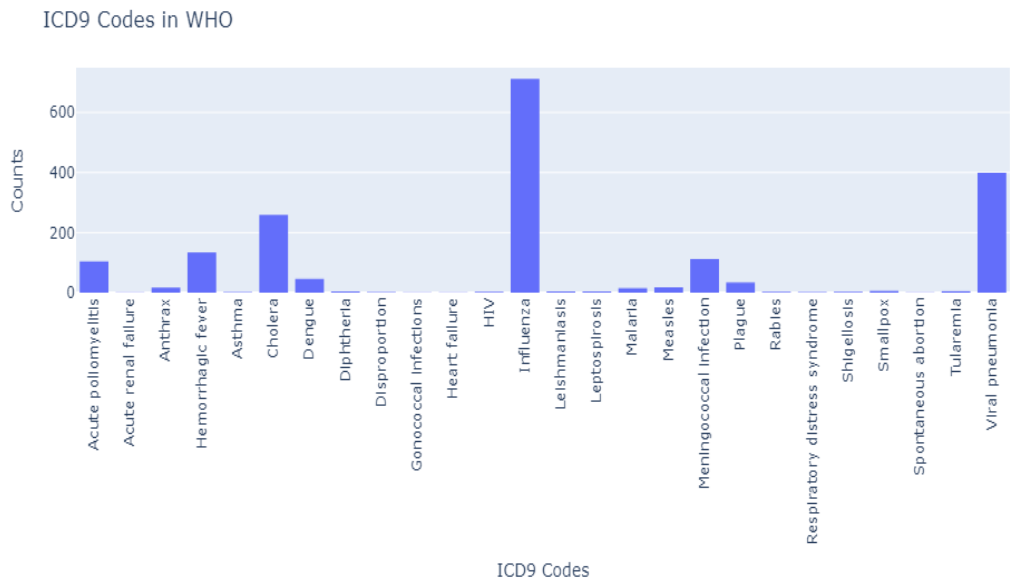

Figure 4.8: ICD9 codes distribution in the generated news dataset with *perfect matching plus* strategy

Note that the generated dataset, especially with respect to the MIMIC-III dataset presented in section 3.2.3, consists entirely of news, i.e. unstructured but fluent text reporting on diseases and medical information in a specific but not too technical language. Moreover, 2302 of the 2504 news are linked to only one ICD9 code, which solves some problems related to the classification task, which can be formulated as a simple classification problem instead of a multi-label classification one on MIMIC-III.

Another remarkable feature of this dataset, that from now on it will be referred to as ***"WHO-ICD9"***, with respect to MIMIC-III is that the majority of the ICD9 codes that appear here are related to communicable disease, which is the target of TrustAlert project. Instead, as exposed in section 3.2.3 referring about *"MIMIC III TOP 10"* or *"MIMIC III TOP 50"*, the most frequent codes in whole MIMIC III are related to uninteresting (for the scope of this project) medical complications such as *Hypertension* or *Cardiac dysrhythmias*.

4.4 Models

This section examines the architectures and training techniques used to solve the problems encountered throughout the thesis considering also the TrustAlert constraints, such as the need for a flexible tagging system that is not constrained by the labels we want to use, or the need to improve performance on specific domains, such as the medical domain.

In section 4.4.1 is presented the standard architecture used to solve multi-label classification problems while highlighting the causes that impede TrustAlert project to fully embrace this kind of solution as a standard.

In section 4.4.2 is presented the chosen architecture, with an interchangeable backbone and fine tuned with a contrastive learning strategy.

4.4.1 Multi-label classifier

A **multi-label classifier** is a machine learning model designed to assign multiple labels to a given input instance. Formally, it can be defined as follows:

- Let $\mathcal{X}$ be the input space, where each instance $x \in \mathcal{X}$ is a text.
- Let $\mathcal{Y}$ be the set of possible labels extracted from one of the available taxonomies. It is represented by $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$, where m is the number of labels.
- The multi-label classifier is a function $f : \mathcal{X} \to 2^{\mathcal{Y}}$, where $2^{\mathcal{Y}}$ denotes the power set of $\mathcal{Y}$.
- The set of parameters θ influences the behaviour of the classifier model $f(x \mid \theta)$. The goal of training is to find a set of parameters θ^* that produces the most effective model $f(x \mid \theta^*)$.

  (effectiveness is not a strictly defined concept but is codified in the structure of the loss function that is chosen based on the desired model behaviour).

For an instance $x \in \mathcal{X}$, the classifier assigns a subset of labels $\mathbf{y} \subseteq \mathcal{Y}$. Thus, the output of the classifier for an input x is a set of labels:

$$f(x \mid \theta^*) = \{y_i \in \mathcal{Y} \mid x \text{ is predicted to have label } y_i\}$$

Since the multi-label classification framework seems to be a natural fit for solving the ICD9 assignment problem, the following section describes how to model and train the architecture according to these principles.

Architecture Design

Working on a NLP scenario, it is preferable to use a model $f(x \mid \theta)$ able to encode and classify textual data. The most common choice, as explained in previous chapter in section 2.3, is to use a *Transformer Encoder* architecture able to generate an embedding vector from the text and then use this vector as input to a classification head *clf*.

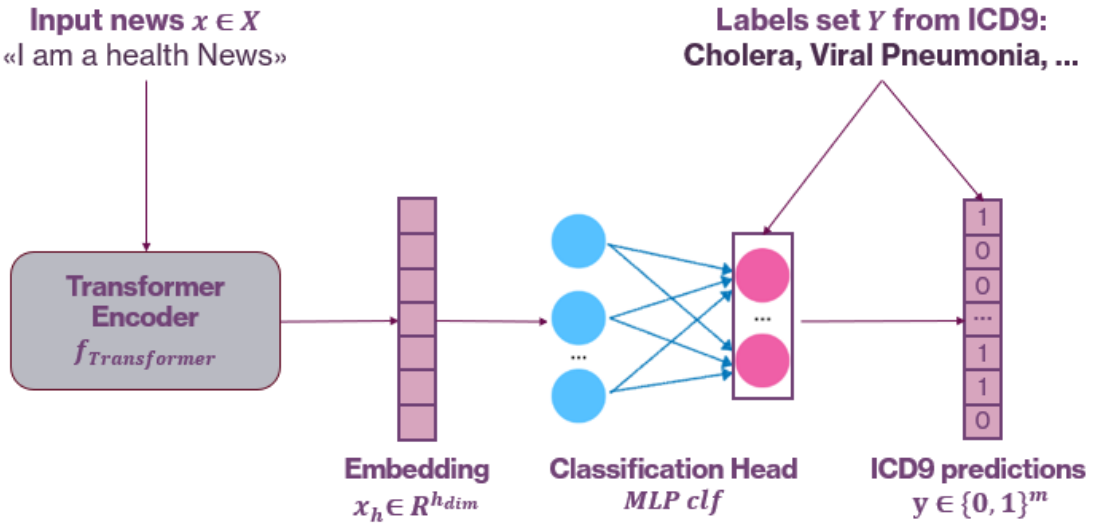

Figure 4.9: Architecture and inference workflow of Multi Label Classifier.

The main components of the architecture $f(x \mid \theta)$, shown in Fig. 4.9 and built to solve the multi-label classification task, are the following:

- **Transformer Encoder**: this could be any model $f_{Transformer}(x)$ that takes in input a textual sequence $x \in \mathcal{X}$ and project it to a vector x_h in a latent space of dimension h_{dim} denoted as $\mathcal{R}^{h_{dim}}$.

$$f_{Transformer} : \mathcal{X} \to \mathcal{R}^{h_{dim}}$$
$$f_{Transformer}(x) = x_h \in \mathcal{R}^{h_{dim}}$$

  The Transformer models used will be *BioBERT*, a variation of *BERT* presented in section 2.3, and *MPNET*, presented in section 2.3.

- **Classification Head**: this could be any model $clf(x_h)$ that takes in input the embedding vector generated by the *Transformer Encoder* $x_h \in \mathcal{R}^{h_{dim}}$ and classify it using a fixed set of labels $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$, where m is the number of labels.

Being a multi-label classification task this classification head must be able to assign a variable amount of labels to each embedding vector x_h, therefore any output $\hat{y}$ is an element of the power set of all possible labels $2^{\mathcal{Y}}$

$$clf(x_h) : \mathcal{R}^{h_{dim}} \rightarrow 2^{\mathcal{Y}}$$
$$clf(x_h) = \hat{y} \in 2^{\mathcal{Y}}$$

The model clf is a multi layer perceptron (or *MLP*) or a single *Linear Layer* that has, as final layer dimension, as many neurons as the possible labels in $\mathcal{Y}$. A threshold is defined on the output neurons and if the relative output is above the threshold the corresponding label is considered to be chosen by the model.

Therefore, instead of working on the powerset of the labels $\mathcal{Y}$, the classifier clf works directly on the set of labels $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$ assigning to each one {0,1} whether they are assigned or not based on the threshold.

$$clf : \mathcal{R}^{h_{dim}} \rightarrow \{0,1\}^m$$
$$clf(x_h) = \hat{y} \in \{0,1\}^m$$

Training Strategy

The training strategies associated to this kind of architectures is divided in two main categories: freezing the backbone model and training only the classification head, therefore maintaining fixed the embedding representation of the *Transformer Encoder*, or training the whole architecture. Effects of both are shown in Fig 4.10.

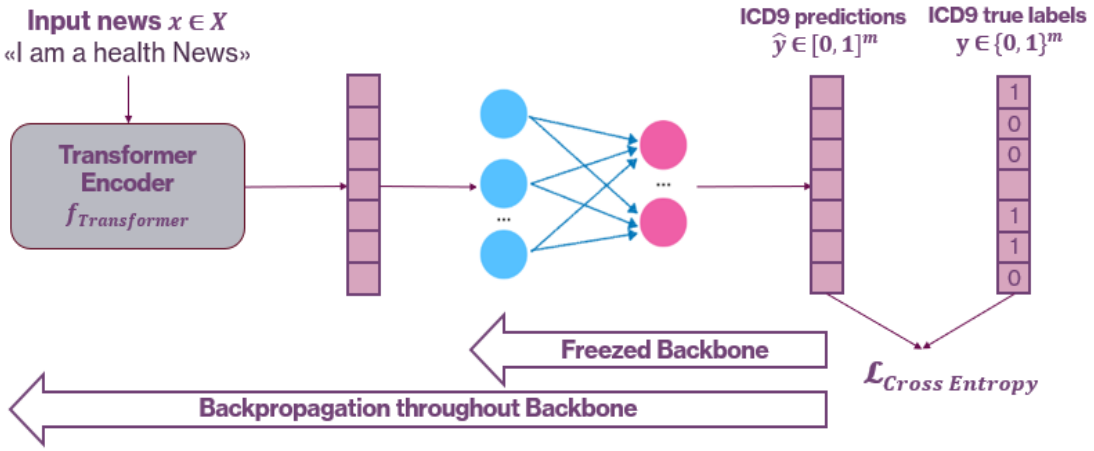

Figure 4.10: Gradient flow through backpropagation at train time. **Above**: update only the *MLP clf*. **Below**: update also the *Transformer Encoder*.

Some important considerations regarding these two approaches are that the number of parameters of the *Transformer Encoder* are way more than the *clf* ones. Therefore, the convergence of training is slower and the amount of data needed is bigger. Moreover, the pretraining of the backbones models are heavy tasks (both in terms of data, resources and time required), can lead to overfit and leveraging upon someone else developed foundation model is becoming the standard behaviour in this field.

Whichever the technique is used, the training task can be formalized as follows.

The multi-label classifier at train time $f(x \mid \theta) : \mathcal{X} \to [0,1]^m$, given a news $x \in \mathcal{X}$, outputs a probability vector $\hat{y} \in [0,1]^m$ that contains at each entry $\hat{y}_i$ the probability of each label being part of the real labels associated to x. Notice that the prediction space $\hat{\mathcal{Y}}$ is $[0,1]^m \subseteq \mathcal{R}^m$ where m is the number of different labels. The true label vector for an instance x is denoted by $\mathbf{y} = (y_1, y_2, \ldots, y_m)$, where $y_i \in \{0, 1\}$.

The loss function used to train is the *cross-entropy* loss, for a single instance $(x, \mathbf{y})$ it is defined as:

$$\mathcal{L}(\ f(x \mid \theta)\ ,\ \mathbf{y}\ ) = -\sum_{i=1}^{m} [y_i \log f_i(x \mid \theta) + (1 - y_i) \log(1 - f_i(x \mid \theta))]$$

Therefore, the objective defined by $\mathcal{L}$ is to align the output label distribution $\hat{\mathcal{Y}}$ with the one we have at our disposal in the training dataset $\mathcal{Y}$.

To minimize the loss over the entire training set $\mathcal{D}$ the total loss $\mathcal{L}_{\text{total}}$ as the average loss over all training examples is defined as:

$$\mathcal{D} = \{(x^{(1)}, \mathbf{y}^{(1)}), (x^{(2)}, \mathbf{y}^{(2)}), \ldots, (x^{(N)}, \mathbf{y}^{(N)})\}$$

$$\mathcal{L}_{\text{total}} = \frac{1}{N} \sum_{j=1}^{N} \mathcal{L}(\ f(x^{(j)} \mid \theta)\ ,\ \mathbf{y}^{(j)}\ )$$

The objective is to find the optimal parameters θ^* such that the multi label classifier $f(x \mid \theta)$ minimizes the total loss:

$$\theta^* = \arg\min_\theta \mathcal{L}_{\text{total}}$$

The optimization is performed through gradient-based optimization methods such as Stochastic Gradient Descent (SGD) or a variation of Adam optimizer [40] like AdamW [41]. Generalizing, the parameters θ are updated iteratively using the gradient of the loss function with respect to the parameters scaled by a learning rate η:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{total}}$$

Problems of multi label classification

The formulation of the tagging problem as a multi label classification task brings some problems already highlighted in the introduction of this chapter 4.

First of all, the structure of the classification head impose to know beforehand both the total number and types of possible labels $\mathcal{Y}$. This means that once a model is developed through this kind of approach, it is bonded to the label set it was trained on. To think about the direct mapping between output neurons and label ordering (which output neuron corresponds to which label) is enough to confirm that it is impossible to adopt this model in an evolving scenario. However, this model should still be able to perform well under a fixed scenario where the focus is specializing the model on a well-defined task.

Another problem is that multi-label classification is clearly a supervised task. This means that we need a dataset $\mathcal{D}$ with a list of ICD9 codes $\mathbf{y} \in \mathcal{Y}$ related to the text sample $x \in \mathcal{X}$ to train the model on. On the one hand, this does not fit well into the environment of the dataset, where MIMIC-III is the only reliable source for this type of data, once again highlighting the lack of such referencing data. On the other hand, this is likely to cause problems related to a shift in data distribution between the training data (medical notes from MIMIC-III) and the data to which the model is applied (news originating from GDELT). This shift can be seen in the average length as well as in the differences in tone, lexical choice and specificity of words belonging to a hospital vocabulary.

As will be seen in the chapter 5, another limitation is that the difficulty of the task increases as the number of labels increases. For the ICD9 tagging task, we have 922 different codes as possible labels (restricting ourselves only to the primary codes). Moreover, the only available labelled dataset for IDC9 codes, namely MIMIC-III described in section 3.2.3, shows a strong imbalance of these classes, and the over represented ones are not even the ones correlated with communicable disease outbreaks that the TrustAlert project wants to focus on. In fact, TrustAlert aims to focus on communicable diseases in order to detect them early and inform specialised structures to prepare for them.

4.4.2 Tagger

The **tagger** is a machine learning model designed to retrieve the topN most similar labels with respect to the semantic content of a text.

- Let $\mathcal{X}$ be the input space, where each instance $x \in \mathcal{X}$ is a text.
- Let $\mathcal{Y}$ be the set of possible labels extracted from one of the available taxonomies. It is represented by $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$, where m is the number of labels.
- The tagger is a function $f : \mathcal{X} \to \{y_i\}^N$, where $y_i \in \mathcal{Y}$ and N is the max number of labels we want to retrieve from all the labels.
- The set of parameters θ influences the behaviour of the tagger $f(x \mid \theta)$. The goal of training is to find a set of parameters θ^* that produces the most effective model $f(x \mid \theta^*)$.

  (effectiveness is not a strictly defined concept but is codified in the structure of the loss function that is chosen based on the wanted model behaviour).

For an instance $x \in \mathcal{X}$, the tagger assigns a subset of N labels $\mathbf{y} \subseteq \mathcal{Y}$. Thus, the output of the tagger for an input x is a set of labels:

$$f(x \mid \theta^*) = \{y_i \in \mathcal{Y} \mid y_i \text{ is in the top N most similar labels w.r.t. } x\}$$

("most similar" is not a strictly defined concept but is codified in the architecture choices such as which Transformer Encoder is used and which distance or similarity metric is adopted).

The formulation presented above is much more similar to a task of information retrieval rather than a classification one. Here, the focus is on finding a way to represent closer the news and their relative labels instead of learning to classify the text in a direct way.

Architecture Design

As specified for the previous model, working on a NLP scenario forces the model $f(x \mid \theta)$ to be able to encode textual data. As explained in section 2.3, the most common approach (section 2.3) is to use a *Transformer Encoder* architecture able to generate an embedding vector from the text. This embedding vector should have a strong relation with the semantic concept present in the text because of the pre training strategies adopted to train these LLMs.

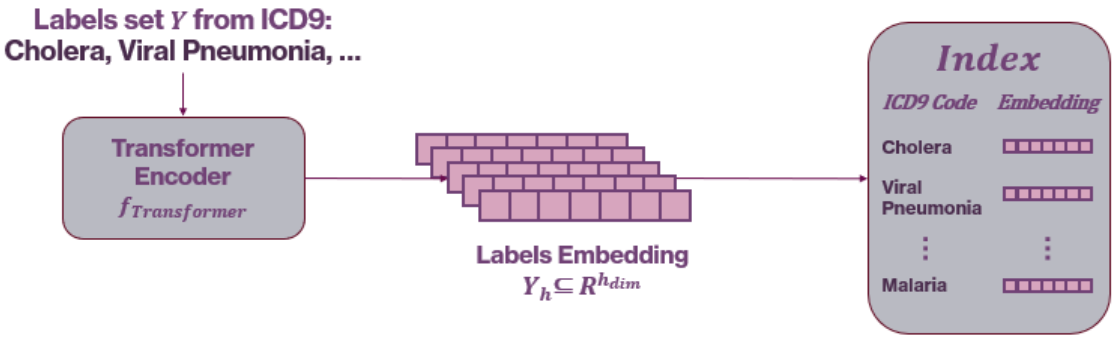

Figure 4.11: Index generation for retrieval task.

The main idea is to fix a *Transformer Encoder* as backbone and generate the latent representation or embedding vector of each label and index them to let a fast retrieval possible, as shown in Fig 4.11. Then, for each text generate its embedding vector and query the index with a specified distance metric such as Euclidean Distance or Cosine Similarity, as shown in Fig 4.12.

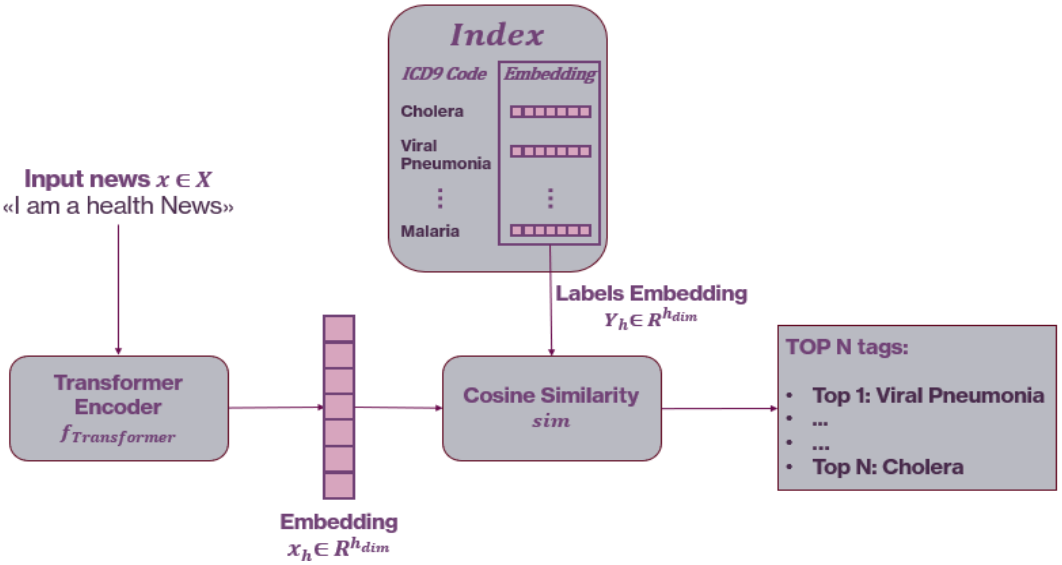

Figure 4.12: Architecture and inference workflow of Tagger.

Therefore, the main parts of the architecture $f(x \mid \theta)$ built to solve the tagging task are:

- **Transformer Encoder**: this could be any model $f_{Transformer}(x)$ that takes in input a textual sequence $x \in \mathcal{X}$ and project it to a vector $x_h \in \mathcal{X}_h$ in a latent space of dimension h_{dim} denoted as $\mathcal{R}^{h_{dim}}$.

$$f_{Transformer} : \mathcal{X} \to \mathcal{R}^{h_{dim}}$$
$$f_{Transformer}(x) = x_h \in \mathcal{R}^{h_{dim}}$$

The Transformer models used will be *BioBERT*, a variation of *BERT* presented in section 2.3, and *MPNET*, presented in section 2.3.

- **Taxonomy**: this time the taxonomy expansion technique used will play a big role. In fact, as presented in section 4.2, using the labels to generate embedding means that their semantic content will play a significant role in shaping their latent representation and consequently the techniques used to generate labels or tags from the taxonomy must be investigated.

$$f_{Transformer} : \mathcal{Y} \to \mathcal{R}^{h_{dim}}$$
$$f_{Transformer}(y) = y_h \in \mathcal{R}^{h_{dim}}$$

The same *Transformer Encoder* used to encode textual data $x \in \mathcal{X}$ is also used to encode labels or tags $y \in \mathcal{Y}$. The set of embedding generated by encoding $y \in \mathcal{Y}$ will be referred to as $\mathcal{Y}_h$. Thus, $y_h \in \mathcal{Y}_h$.

- **Index**: this is a data structure useful to store and quickly retrieve the embedding of the labels $y_h \in \mathcal{Y}_h \subseteq \mathcal{R}^{h_{dim}}$. Given the embedding of a text as a query vector $x_h \in \mathcal{X}_h \subseteq \mathcal{R}^{h_{dim}}$, this index retrieves the N nearest labels in the latent space $\{y_{h_i}\}^N$ where $y_{h_i} \in \mathcal{Y}_h$ with respect to $sim(x_h, y_h)$, namely a *distance metric* or *similarity metric*.

$$sim : \mathcal{X}_h \times \mathcal{Y}_h \to \mathcal{R}$$
$$Index : \mathcal{X}_h \to \{y_{h_i}\}^N$$
$$Index(x_h \,|\, \mathcal{Y}_h \,,\, sim) = \arg\max_{i \in \mathcal{Y}_h} \{sim(\, x_h \,,\, y_{h_i}\,)\}$$

(If a distance metric is used instead of a similarity one, the formulation of the Index will use an $\arg\min$ *instead of an* $\arg\max$*)*

Training Strategy

The architecture presented is quite unusual with respect to the most common ones for text classification. However, some similarities can be found if we look at *Retrieval Systems*, *Siamese Networks* [42] [29] [43] and at *contrastive learning* frameworks [44] [45] [46] [47]. Usually, *Retrieval Systems* are trained with adversarial or contrastive learning strategy because the focus is on developing a latent space that resemble the distances between concepts that we want to encode.

Therefore, to train this kind of architecture the idea explored is the one of *contrastive learning*.

Initially introduced for computer vision tasks [44], the main idea is to force the latent representation of pairs of similar images to be near each other in the latent space and separating the representation of the pairs of different images, this behaviour is visualized in Fig 4.13. The concept of "similar images" can be easily represented by images that belong to the same class, or taken geographically at the same coordinates. This idea is represented by the *Contrastive Loss* [48] $\mathcal{L}(x_i, x_j)$:

$$\mathcal{L}_{Contrastive}(x_i\ ,\ x_j) := \begin{cases} d(x_i, x_j) & \text{if } \mathbf{y_i} == \mathbf{y_j} \\ \max\{0, m_n - d(x_i, x_j)\} & \text{if } \mathbf{y_i} \neq \mathbf{y_j} \end{cases}$$

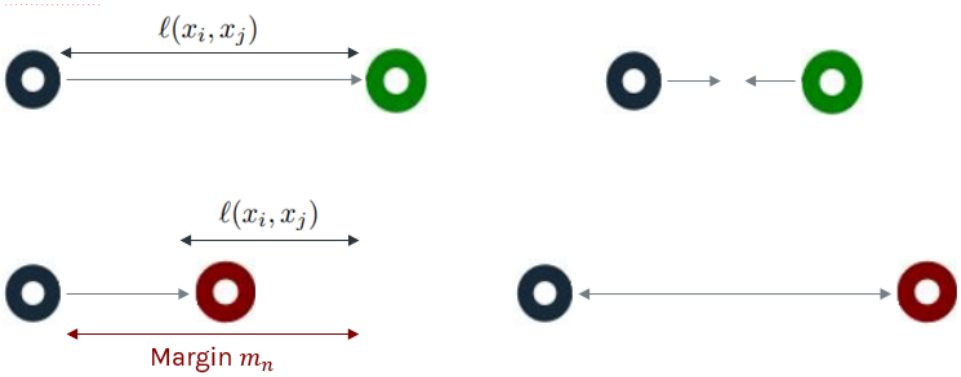

Figure 4.13: Contrastive Loss visual representation. **Above:** the images representations of a positive pair are pulled together in the latent space. **Below:** the images representations of a negative pair are pushed away in the latent space enforcing the margin m_n.

Some more refined techniques such as *Triplet Loss* [31] and *Multisimilarity Loss* [49] introduce the idea of bringing together the positive images while keeping apart at the same time the images that do not belong to the same class.

The point to focus on is the "at the same time". In fact, the main difference is that these losses do not work with pairs but triplets or entire groups. The

representation of the image is simultaneously brought closer to the similar sample and away from the different ones. This behaviour is visualized in Fig 4.14.

Following the standard nomenclature the image is defined as *anchor "a"*, the similar image is called *positive "p"* and the different one is called *negative "n"*.

Given an anchor a, a positive sample p, and a negative sample n, and a margin m, the triplet margin loss $\mathcal{L}_{Triplet}$ is defined, using a *distance metric* $d(x\,,\,y)$ or a *similarity metric* $sim(x\,,\,y)$, as:

$$\mathcal{L}_{Triplet}(a, p, n) = \max\left(0, d(a, p) - d(a, n) + m\right)$$

or

$$\mathcal{L}_{Triplet}(a, p, n) = \max\left(0, sim(a, n) - sim(a, p) + m\right)$$

Figure 4.14: Triplet Loss visual representation. The image representations of the positive p and the anchor a are pulled together in the latent space while the representation of the negative n and the anchor one a are pushed away enforcing the margin m_n.

Triplet Dataset Structure

The triplet loss can be adapted to the NLP scenario with the specific intent of developing a training strategy for the *Tagger*. To do so, it is mandatory to structure a dataset $\mathcal{D}$ of triplets of data $(x, \mathbf{y_p}, \mathbf{y_n})$. Taking the second level of tagging (section 4.1.2) as an example, for each news x two different labels are provided, both extracted from the ICD9 taxonomy. The positive label $\mathbf{y_p}$ must be one of the diseases the news x is talking about and the negative label $\mathbf{y_n}$ can be sampled at random among all the remaining codes.

$$\mathcal{D} = \{(x^{(1)}, \mathbf{y_p}^{(1)}, \mathbf{y_n}^{(1)}) \, , \, \dots \, , \, (x^{(N)}, \mathbf{y_p}^{(N)}, \mathbf{y_n}^{(N)})\}$$

Using dataset $\mathcal{D}$, the triplet loss $\mathcal{L}_{Triplet}(a, p, n)$ can be easily adapted using the *Transformer Encoder* $f_{Transformer}(x|\theta)$ like in a *siamese network* setting [29].

Given the triplet $(x, \mathbf{y_p}, \mathbf{y_n})$ are processed by the model with the same set of parameters θ as follows:

- ***anchor "a"*** is generated from the news x as $f(x \,|\, \theta)$.
- ***positive "p"*** is generated from the positive label $\mathbf{y_p}$ as $f(\mathbf{y_p} \,|\, \theta)$.
- ***negative "n"*** is generated from the negative sampled label $\mathbf{y_n}$ as $f(\mathbf{y_n} \,|\, \theta)$.

To compute the triplet loss the similarity chosen is the *Cosine Similarity* between the embedding generated.

$$\mathcal{L}_{Triplet} : \mathcal{X}_h \times \mathcal{Y}_h \times \mathcal{Y}_h \to \mathcal{R}$$

$$sim(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

$$\mathcal{L}_{Triplet}(a, p, n) = \max\left(0, sim(a, n) - sim(a, p) + m\right)$$

$$\mathcal{L}_{\text{total}} = \frac{1}{N} \sum_{j=1}^{N} \mathcal{L}_{Triplet}(\ f(x^{(i)} \,|\, \theta) \ , \ f(\mathbf{y_p}^{(i)} \,|\, \theta) \ , \ f(\mathbf{y_n}^{(i)} \,|\, \theta) \ )$$

The objective is to find the optimal parameters θ^* such that $f_{Transformer}(x|\theta^*)$ minimizes the total loss $\mathcal{L}_{\text{total}}$.

Considerations over Tagger architecture

The *Tagger* architecture proposed brings several advantages with respect to the multi-label classifier.

First of all, the label set $\mathcal{Y}$ is no more a hard constraint of the model, and this means that can be easily kept up to date with the diseases and medical landscape. For example, this model can work with any version of the ICD taxonomy past, present and future ones.

Another advantage is that the backbone model, an LLM such as *MPNET*, is basically used only as an encoder and therefore any progress done in NLP field can be easily applied to this design. Thus, this architecture is also easier to update technologically speaking.

The only remaining problems reside in the definition of a Dataset $\mathcal{D}$ given that the *Contrastive Learning* is still a supervised framework (or self-supervised in some cases). However, any pre trained model can be used in a zero-shot scenario with no further fine-tuned with this architecture design. In this case, an evaluation suite is mandatory to be able to assess its capabilities.

Chapter 5

Experiments and Results

This chapter reports on all the experiments that were considered relevant to support the statements within the discussion (chapter 6) of the thesis. This chapter is divided by grouping the experiments into those that evaluate the *"phase one"* tagging, which filters news based on correlation with "health" topics, and those that evaluate the *"phase two"* tagging, which tags the news with the corresponding ICD9 code.

As far as the experiments for the first phase are concerned, the dataset involved is the one generated specifically for this thesis and presented in section 4.3.1.

The section about the second phase focuses instead on comparing the two strategies explained in chapter 4: *multi label classification* (4.4.1) and *tagging* (4.4.2). The datasets used for this group of experiments are MIMIC-III (3.2.3) and "WHO-ICD9", the news dataset adapted for ICD9 tagging (4.3.1).

Given the similarities between the tagging strategy and the retrieval task, the metrics chosen for the task evaluation belong to the retrieval system framework. In order to fully understand the results proposed, section 5.1 explains all the evaluation metrics encountered during this chapter.

5.1 Metrics of Evaluation

RETRIEVAL METRIC

Inspired by *Retrieval Systems* evaluation based on *metrics@N*, where it is possible to decide how many of the retrieved labels are considered meaningful, it is introduced the *Accuracy@1*.

Accuracy@1: this evaluation metric is computed looking whether the Top 1 label retrieved by the tagger $\hat{y} \in \hat{\mathcal{Y}}$ is in fact the label associated with the input data $(x\,,\,y)$, where $y \in \mathcal{Y}$, or it is in the set of labels associated with the data $(x\,,\,\mathbf{y})$, where $\mathbf{y} = \{y_i\,,\,...\,,\,y_m\}$ and $y_i \in \mathcal{Y}$ and $\mathbf{y} \in \mathbf{Y}$.

$$\textbf{acc@1} : \hat{\mathcal{Y}} \times \mathbf{Y} \rightarrow \mathcal{R}$$

$$\textbf{acc@1}(\hat{y}\,,\,\mathbf{y}) = \begin{cases} 1 & \text{if } \exists y \in \mathbf{y} : \hat{y} = y \\ 0 & otherwise \end{cases}$$

$$\textbf{Accuracy@1}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} acc@1(\, f(x^{(i)} \,|\, \theta),\, \mathbf{y}^{(i)})$$

ClASSIFICATION METRICS

Precision: as the ratio between the number of news correctly classified as class y and the total number of news classified as y.

$$\textbf{Precision} = \frac{TP}{TP + FP} \tag{5.1}$$

Where TP is the number of *True Positives*, the ones correctly classified as $\hat{y}$, and FP is the number of *False Positives*, those that were erroneously classified as $\hat{y}$, e.g. as news reporting on *"health"*, when they were about something else.

Recall: as the ratio between the number of data points correctly classified as y and the total number of news actually belonging to the class y.

$$\textbf{Recall} = \frac{TP}{TP + FN} \tag{5.2}$$

Where FN is the number of *False Negatives*, the news which have been erroneously assigned to other classes, when they are actually of the class under examination.

F1: an harmonic mean between Precision and Recall

$$\textbf{F1} = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{5.3}$$

5.2 Tagger Phase One

In this group of experiments it is evaluated how the first level tagger, introduced in section 4.1.1, behave on a dataset of news. It must distinguish between news related to health topic and news that talk about anything else. Therefore, the evaluation setting is very similar to a binary classification problem with two classes *"health"* and *"not health"*. However, as profoundly explained in section 4.4, the model used is a *Tagger*. Thus, the output of the model is a ranked list of labels ordered by proximity in the latent space with respect to the input news.

The datasets used to evaluate the first setting is the one developed for this thesis project and presetned in section 4.3.1.

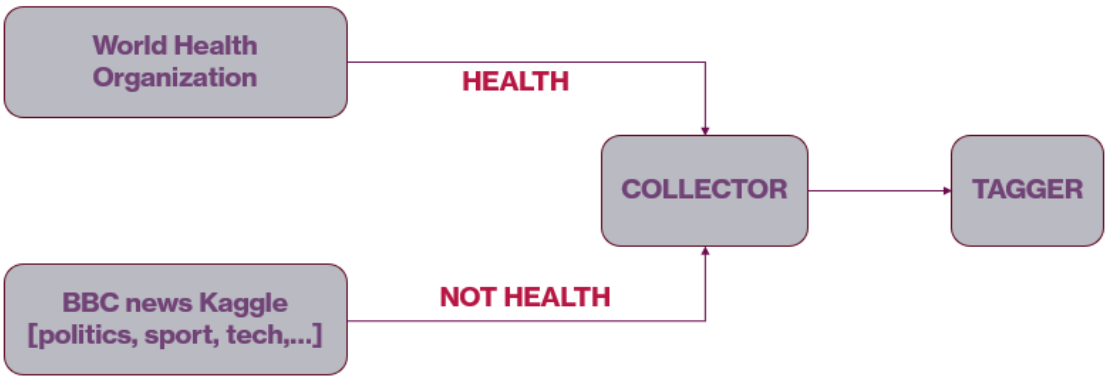

Figure 5.1: A specific module *Collector* is responsible to collect the news from *WHO* API and *BBC* Kaggle dataset generating the dataset presented in section 4.3.1.

5.2.1 Base Model Performances

The first experiment has the goal of assessing how the model chosen behave on the dataset without any further training or fine-tuning. The backbone model used for this experiment is an *MPNET*, explained in section 2.3. The intended behaviour is classifying all the news coming from *WHO* dataset as *"health"* and all the ones belonging to *BBC* as *"not health"*.

Model	***Dataset***	***Accuracy @ 1***
MPNET	**WHO**	18.0
MPNET	**BBC**	98.96

Table 5.1: Accuracy@1 of Tagger on the two separated datasets.

In Table 5.1 it is possible to see how the model performs in distinguishing

between *"health"* related and *"not health"* related news. The immediate conclusion is that, while the model is quite good at understanding what it is not related to *"health"*, given the 98.96 on *BBC* dataset, it is not nearly as good at identifying what it is related to *"health"*. In fact, **only 18.0 on *WHO* dataset means that 82% of news in *WHO* is misclassified**, therefore tagged with tags different from *"health"*.

The main hypothesis that was proposed and verified by the experiments in the following section is that the only word *"health"* it is not very semantically meaningful. Therefore, there is the need to enrich the label semantic content.

5.2.2 Taxonomy Expansion Techniques Comparison

This set of experiments is designed to asses the differences brought by changing the semantic content of the labels. Notice that these labels are generated from the *IPTC Taxonomy* (section 3.1.1) and used by *Tagger* to tag the news using mainly their semantic content. In changing it, it is expected a drastic change in the results.

By keeping both the model and the input data the same, only the labels are modified using two specific techniques *"primary"* and *"secondary"* explained in details in section 4.2.

Dataset	***Taxonomy "primary" Accuracy @ 1***	***Taxonomy "secondary" Accuracy @ 1***
WHO	18.0	**97.7**
BBC	98.96	98.56

Table 5.2: Accuracy@1 of Tagger comparing Taxonomy Expansion techniques.

As it was expected, the semantic enrichment of the labels brought a significant improvement. As it is possible to see in Table 5.2, with a mere **-0.40** on *BBC*, the *Tagger* is now fully able to recognize *"health"* related news. This improvement is quantified by a **+79.7** on *WHO* for a total of **97.7** of Accuracy@1.

Looking at the same problem from a binary classification perspective, i.e. considering a news item to be classified as health if and only if the Top1 retrieved tag is *"health"*, it is possible to use metrics common to the classification framework, as shown in Table 5.3 achieving a final F1 score of **0.98**.

Expansion Technique	**Accuracy *@1***	**Precision *"health"***	**Recall *"health"***	**F1 *"health"***
Taxonomy "primary"	0.5384	0.9458	0.1804	0.3029
Taxonomy "secondary"	**0.9811**	**0.9855**	**0.9775**	**0.9815**

Table 5.3: Performance Metrics for news dataset using *MPNET*.

5.2.3 Backbone Model Comparison

Looking mainly for zero-shot performances, it is investigated which model among the pool of openly available and pretrained ones is the best on this news classification task. Once established, thanks to the previous experiment, that the expanded taxonomy with *"secondary"* technique gives more information to the models to work with, all the comparisons reported in Table 5.4 are performed with the *Taxonomy "secondary"*.

Backbone Model	**Accuracy** ***@1***	**Precision** ***"health"***	**Recall** ***"health"***	**F1** ***"health"***
BERT				
[CLS] pooling	0.4457	0.6372	0.0110	0.0217
mean pooling	0.4423	0.0	0.0	0.0
RoBERTa				
[CLS] pooling	0.4429	0.4473	0.0538	0.0960
mean pooling	0.4423	0.0	0.0	0.0
XLNet				
[CLS] pooling	0.4419	0.3882	0.0028	0.0056
mean pooling	0.4423	0.0	0.0	0.0
MiniLM-L6	0.8753	0.9864	0.7850	0.8742
MPNET	**0.9811**	**0.9855**	**0.9775**	**0.9815**
QA-MPNET	0.6308	0.9832	0.3426	0.5081

Table 5.4: Backbone comparison for first level tagging task on news dataset.

The main conclusions that can be drawn from the Table 5.4 are that sentence transformers are better suited for this kind of semantic embedding generation and the best performing one among them is the *MPNET* model. This model is also used as a starting point for second level tagging.

5.3 Tagger Phase Two

In this second group of experiments it is evaluated how the second level tagger, introduced in section 4.1.2, behave on both "WHO-ICD9", the dataset of news adapted to ICD9 tagging, and on MIMIC-III. It must tag each news with the related ICD9 code. Therefore, the evaluation setting is very similar to a multi label classification problem with as many classes as the codes present in the ICD9 taxonomy, presented in 3.1.2. Both the dataset may have a different number of codes associated to each news or discharge note. Thus, while computing retrieval metrics we look whether the top1 tag belong to the set of ICD9 codes used as golden truth.

5.3.1 Dataset Adaptation Techniques Comparison

These experiments aims to understand which version of "WHO-ICD9" is the most suited to be used as test suite. As it is possible to see in Table 5.5, the dataset generated with only *"Perfect Match"* is quite easy to work with. In zero-shot environment and with no taxonomy expansion the result is an accuracy of **0.91**. However promising, the *"Perfect Match Plus"* strategy makes the dataset generated larger and also more diverse in terms of labels, as presented in section 4.3.2. The lower results in zero shot are indeed a good thing, it means that this dataset is now a difficult scenario for our model.

Extraction Method	***Taxonomy Expansion***	***Accuracy@1***
Perfect Match Plus	"Primary"	0.635
Perfect Match Plus	"Secondary"	0.728
Perfect Match Plus	**"Definition"**	**0.740**
Perfect Match	**"Primary"**	**0.913**
Perfect Match	"Secondary"	0.887
Perfect Match	"Definition"	0.856

Table 5.5: Performance Metrics using MPNET for Different Extraction Methods and different taxonomy techniques in adapting news dataset to ICD9 coding.

Table 5.5 explores also the relationship among the *Taxonomy Expansion techniques*. An interesting fact is that while in *"Perfect Match"* the dominant *Taxonomy Expansion technique* is *"Primary"*, when the dataset becomes more difficult, as in *"Perfect Match Plus"*, the strategies that bring better results are *"Secondary"* and *"Definition"*. This means that a description of the diseases helps the model in

understanding the semantic relationship between the labels and the news.

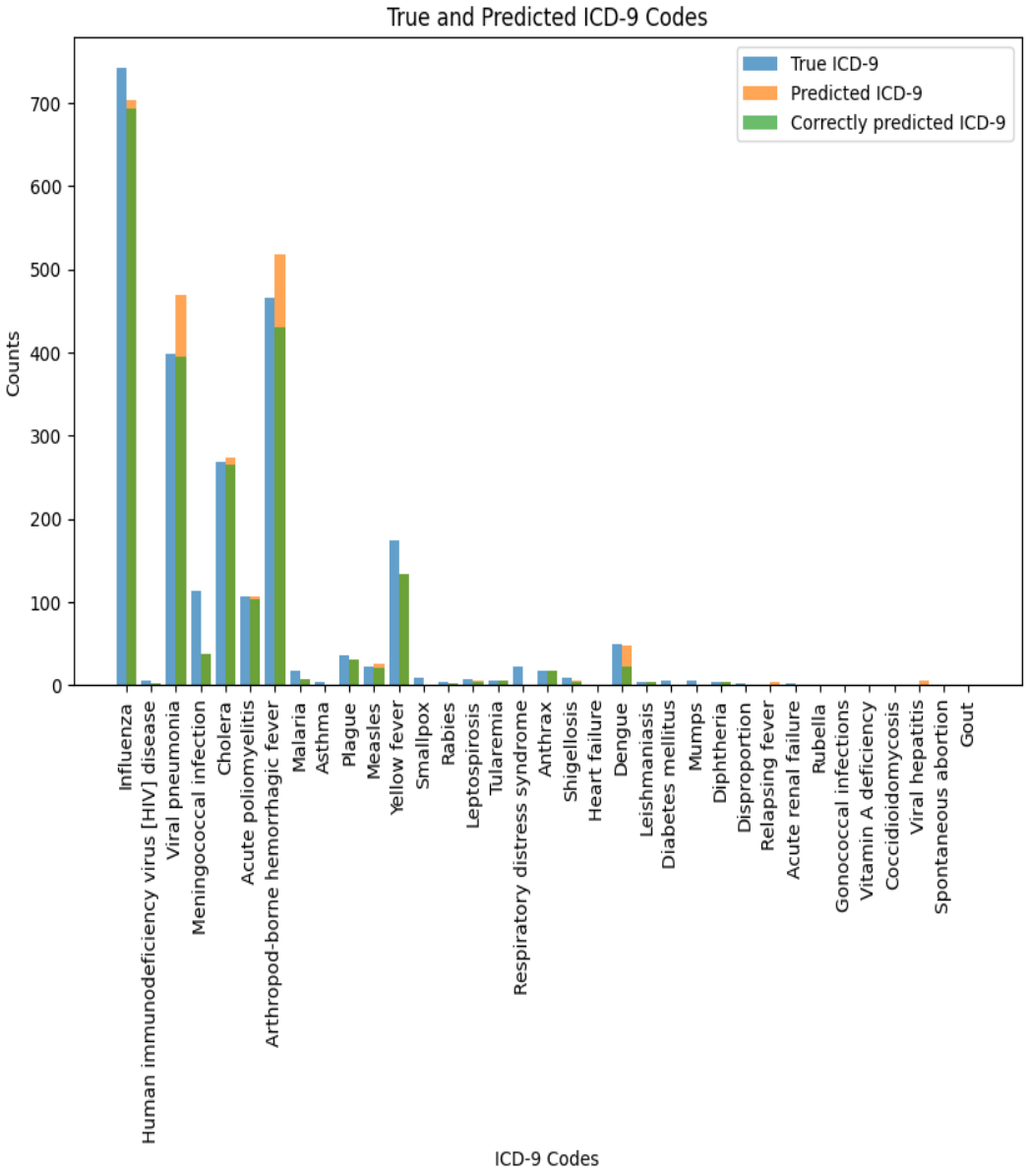

Figure 5.2: Results of ICD9 tagging on *WHO* dataset adapted through *"Perfect Match Plus"*, using MPNET base model.

Figure 5.2 shows how the tagging system responds to the variety of ICD9 codes present in the dataset. It is important to note that while the bars on the left represent the actual number of each specific code, on the right it is possible to see how many of these codes are assigned by the tagger and how many of those are

correctly assigned. Interestingly enough, the behaviour on rare codes is promising

5.3.2 Multi label classification fine-tuning

As introduced in the methodology, within chapter 4, the first approach was to build a *Multi-label classifier* (section 4.4.1) and train it on MIMIC-III (section 3.2.3) to understand if the task was feasible and interesting for the TrustAlert project.

Even considering the architecture limitations profoundly highlighted in the introduction of this approach, the results found remain interesting to discuss.

Multi label classification

In Fig. 5.3 are reported the training and the evaluation metrics on MIMIC-III dataset. Notice that the classifier is trained keeping frozen the backbone model, in this case MPNET base model, and tuning only the classification head. In fact, backpropagating through the backbone gave close to none results.

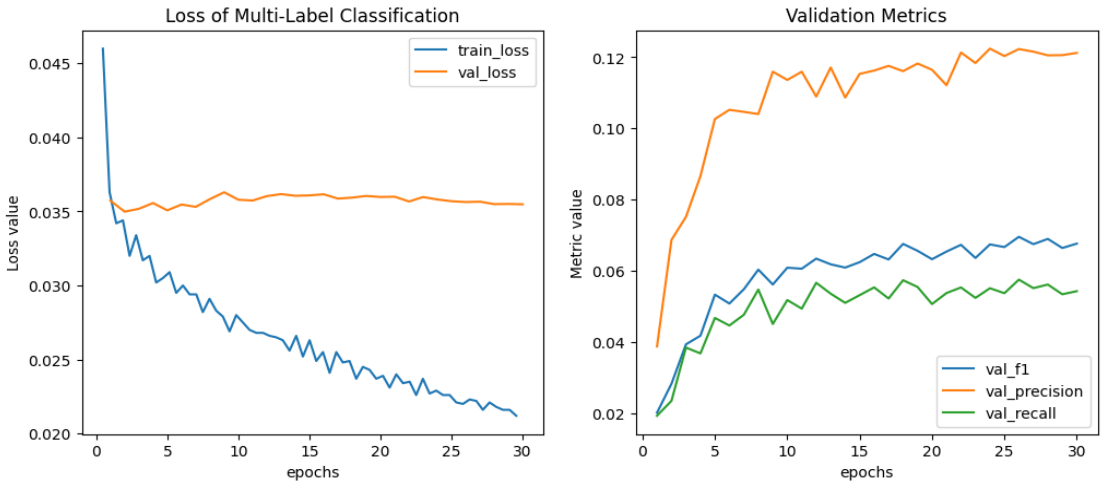

Figure 5.3: Training and evaluation of multi label classification with MPNET on MIMIC-III.

A major problem we can see is represented by the different behaviours on the training and validation sets, in particular a decrease in the train loss is not reflected in one on validation, even if the evaluation metrics continue to increase. Another major problem is the poor final performance achieved by this strategy, only **0.12** in F1-macro. A plausible cause for this lack of effective training could be the inherent difficulty of the MIMIC-III text, given as input data. To evaluate also another model, *BioBERT* was used as backbone too. This model is usually used for tackling scientific and medical text, its behaviour is shown in Fig 5.4.

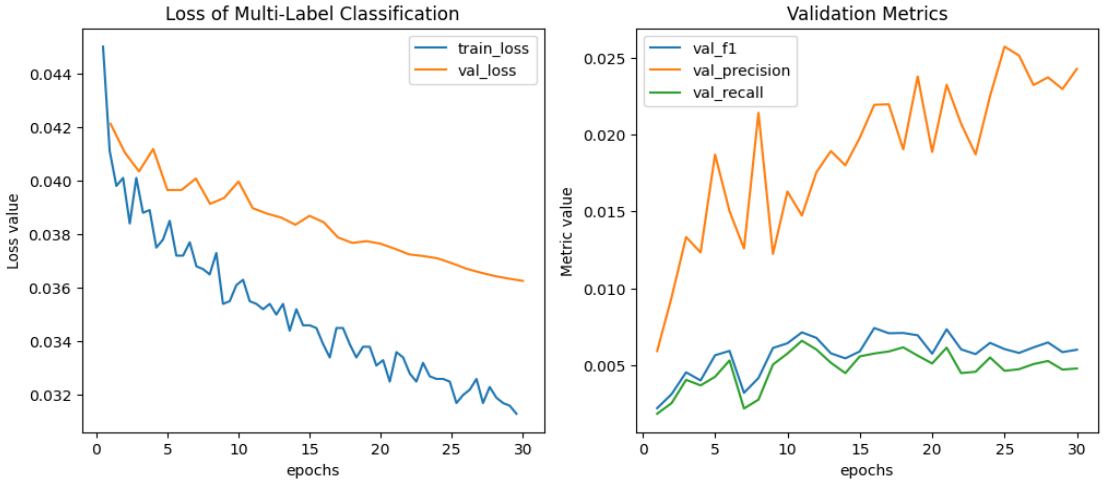

Figure 5.4: Training and evaluation of multi label classification with BioBERT on MIMIC-III.

Even if with *BioBERT* the training losses behave as expected, the evaluation metrics are way more floaty and, with the same training time of 30 epochs, the final results are worst. Once more the *MPNET* is the model that behave the best, as shown in Table 5.6.

Backbone Model	***Loss***	***Precision***	***Recall***	***F1***
MPNET	**0.035**	**0.121**	**0.054**	**0.067**
BioBERT	0.0362	0.024	0.0047	0.006

Table 5.6: Different Model performances as multi label classifier backbone.

Multi label classification as backbone fine-tuning

An idea to explore was to look at the performances of the multi label classification as finetuning strategy for the backbone model. Precisely, the idea was not to freeze the backbone model while training the classifier and look whether the modified model would be better at tagging the news. As easily predictable, the performances of this obtained tagger were heavily degraded. The main reason may be that the multi label classification task, trained in this way, uses the backbone model mainly as a feature extractor, then the classification head loses the relationship with semantics and simply tries to fit the embedding to the correct output neurons. Trying to optimize the model to fit these requirements may be as asking the model to lose semantic capability to fit the constraints of classification. Whereas the tagging task is mainly a retrieval search based on semantics.

MLM finetuning

The final attempt to extract value from MIMIC-III was to use its text to additionally train the *MPNET* with the MLM strategy, to assess if it could benefit the multi-label classification task. To do so, the discharge notes from MIMIC-III were fed into the model with some terms masked: 15% of tokens were replaced with a special token [MASK] and the model tried to predict which word was under the mask.

The MLM training is shown in Fig 5.5 , whereas the final multi label classification task is shown in Fig 5.6.

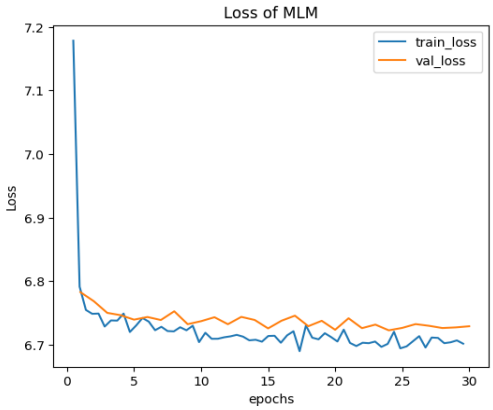

Figure 5.5: Finetuning MPNET with MLM strategy on MIMIC-III.

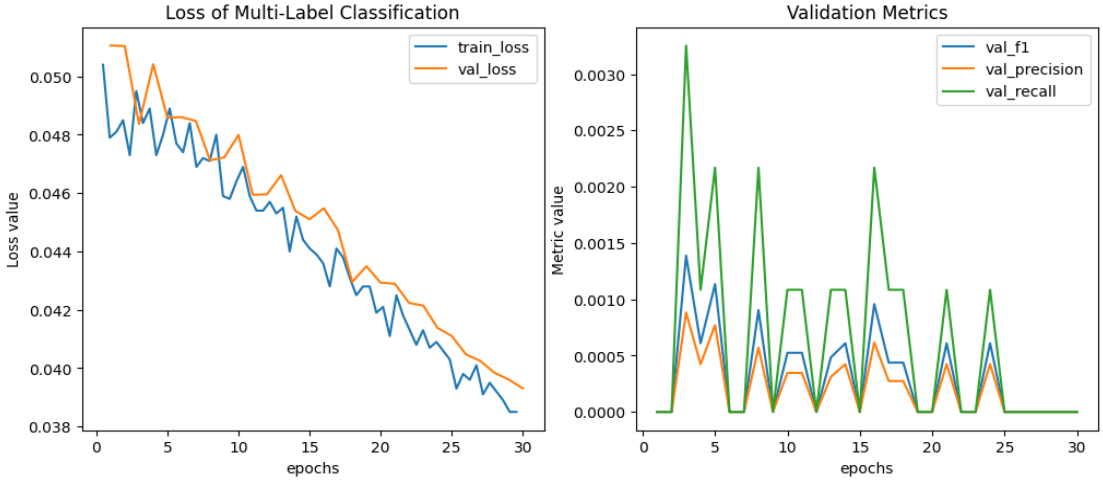

Figure 5.6: Training and evaluation of multi label classification with MPNET with MLM on MIMIC-III.

The main conclusions that can be drawn are the inadequacy of both the MIMIC-III dataset for the multi-label classification task and the structure of multi-label classification with such a large number of possible labels.

Some interesting but unexplored ways to finetune these models could be to work on vocabulary expansion to let the model be more aware of terms that are specific to a domain of interest. Unfortunately, some problems of resources both in terms of computational capabilities and quality data availability impeded to pursue this strategy. However, in section 7.2 some ideas about possible pretraining and finetuneing strategies for semantic domains with specific knowledge are explored.

5.3.3 Contrastive Learning fine-tuning

The approach of the *Tagger* (section 4.4.2) was more promising in terms of future proof architecture and flexibility. In this section are evaluated its capabilities in two different scenarios: on codes seen during training and on codes that the model has never seen. All the training cycles are performed for a duration of 10 epochs, the evaluation is performed once at the beginning and once at the end of the training, using in the former case the base model and on the latter the model trained using contrastive learning. Notice that the labels embedding must be recomputed with the new model parameters and the Index must be recreated.

ICD9 Codes seen during training

For this experiment, the adapted news dataset (section 4.3.1) is split in a stratified way in train set and evaluation set, therefore maintaining the same labels distribution in the two sets. Specifically, in train set we have 1687 different news with 31 distinct ICD9 codes associated to them. In the evaluation set we have 817 different news associated to 24 different ICD9 codes. Both the distribution are shown in Fig 5.7 and in Fig 5.8.

The fine-tuning through contrastive learning is performed on train set and evaluated on codes that it has seen during training. The goal of the experiment is to understand and quantify the imporvments the model can achieve with this strategy of training.

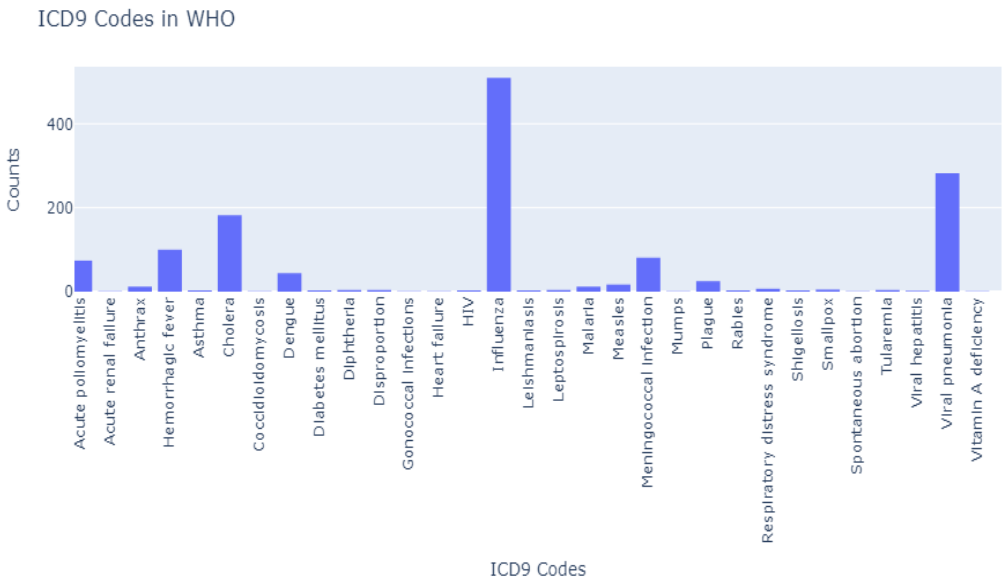

Figure 5.7: Train set label distribution.

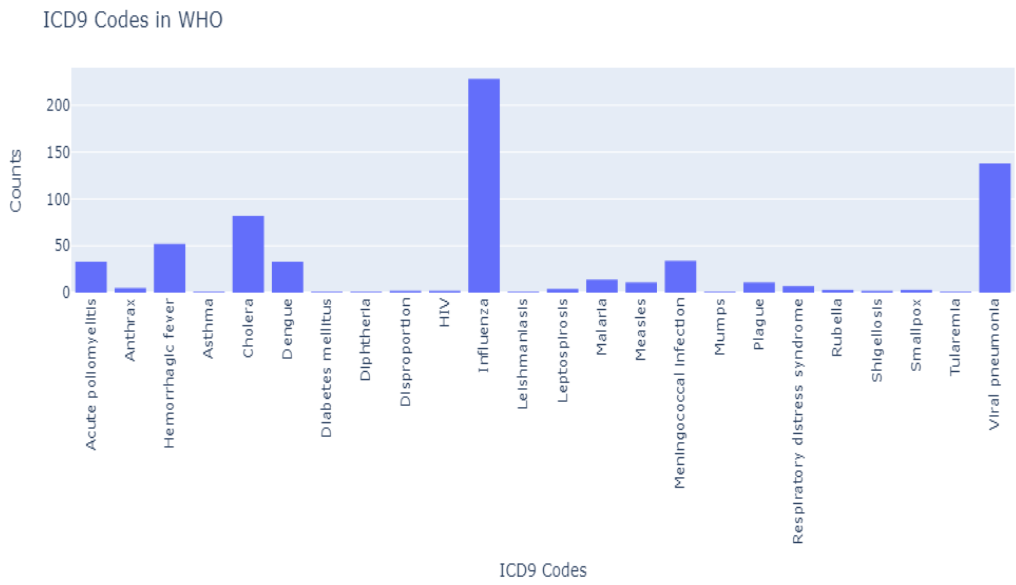

Figure 5.8: Validation set label distribution.

The result brought by this training strategy, reported in Fig 5.9, shows a progress on evaluation data **from 70.50% to 90.70%** in Accuracy@1. Namely, **now 9 out of 10 times the model is able to chose among all the possible ICD9 codes, which is the one the news is talking about**. This highlight the effectiveness of contrastive training.

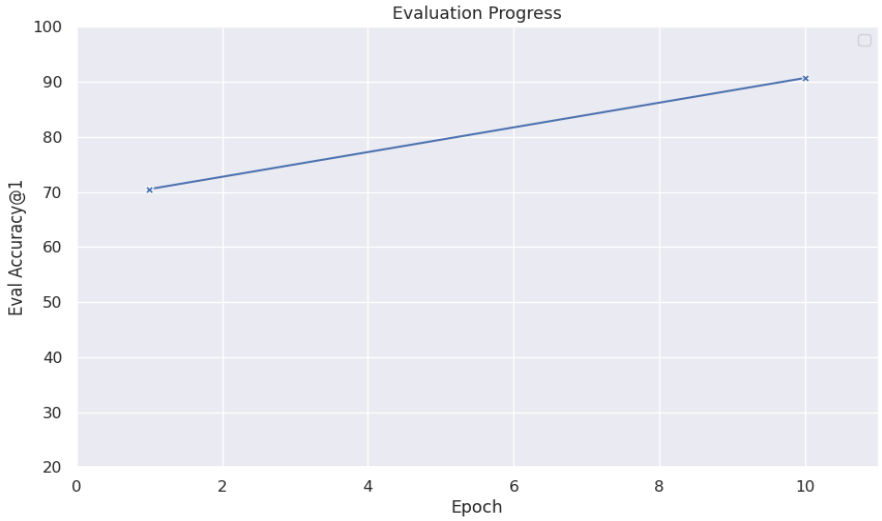

Figure 5.9: Results of contrastive learning.

Zero Shot ICD9 Codes

Due to the scarcity of quality annotated data for ICD9 tagging, the other setting it was important to explore was the zero shot scenario. It is important to verity mainly two things. The first one is whether the model, by never looking at any news about a code, is able to still tag it correctly. The second one is whether the performances of the model on never seen codes is degraded because of the contrastive learning finetuning on other codes. In other words, the check to perform is to asses if the contrastive learning translate into a forgetting of the codes not present in the train set.

To answer to these questions an experiment is performed where a subset of labels considered problematic are kept-out from both train and validation set. This set is referred to as *"Kept-out evaluation"*. The training is again performed on train set and the evaluation is performed at the beginning and at the end on both evaluation sets.

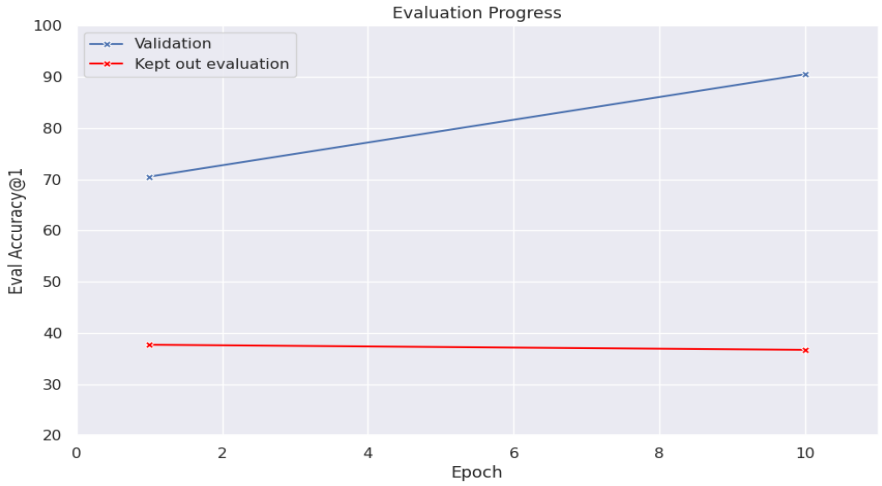

Figure 5.10: Results on both codes seen and never seen during training.

As shown in Fig 5.10, the **evaluation metrics on codes seen during training still improves significantly, as expected, from 84.66% to 94.89% in Accuracy@1**. The **results on *"Kept-out evaluation"* set present a slight decreasing going down from 37.68% to 36.68%**.

Firstly, the higher initial score on the evaluation set is explained by the fact that the excluded codes were selected using the labels on which the model performed worst, making the evaluation set easier.

The results on *"Kept-out evaluation"* show that the model does not generalise further to unseen codes, but the degradation in performance on this type of data does not seem to be too impactful.

MLM additional pretraining

One last experiment important to make was trying the Masked Language Modeling as a pretraining strategy to then apply the contrastive learning strategy. To evaluate this approach on "WHO-ICD9" dataset the chosen strategy was to collect as plain text all the news within the dataset and perform with the backbone transformer encoder of the tagger the MLM strategy. This means masking a percentage of the text (around 15% of tokens in the text sequence) and let the backbone model chose among the vocabulary tokens which one is most suited to fill the *"[MASK]"*.

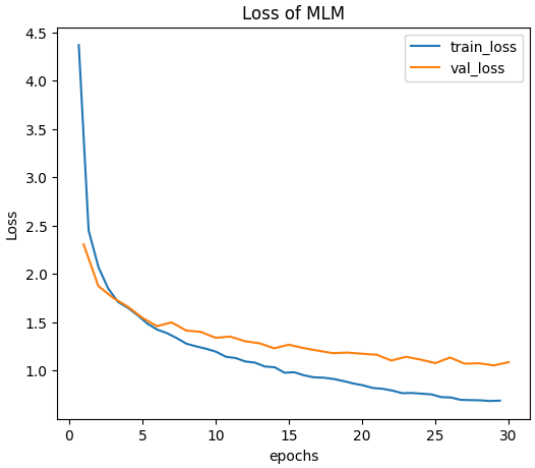

Figure 5.11: MLM pretraining strategy on "WHO-ICD9" text, MPNET backbone used.

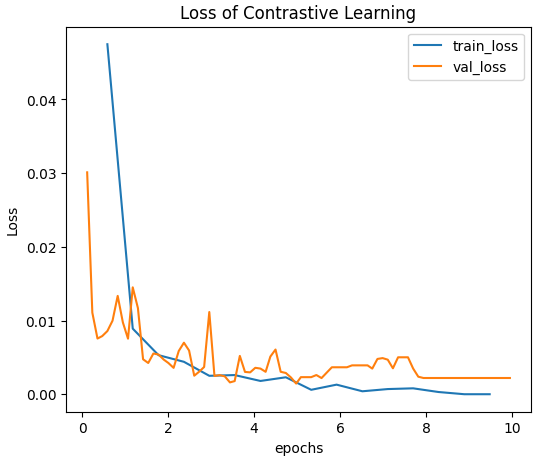

Figure 5.12: Contrastive Learning strategy with best model on MLM pretraing.

As shown in Fig 5.11, during MLM pretraining the *MPNET* base model used as backbone is learning to correctly fill the masked tokens with the terms of its vocabulary. Then, once the 30 epochs are completed, the best model according to the evaluation loss is chosen as backbone for the tagger and the same contrastive learning training is performed. In Fig 5.12 is shown the progress throughout the process of contrastive learning with Triplet Loss and Cosine Similarity as similarity metric.

However effective, is interesting to also compare the progress achieved by this approach with respect to the one without the MLM pretraining.

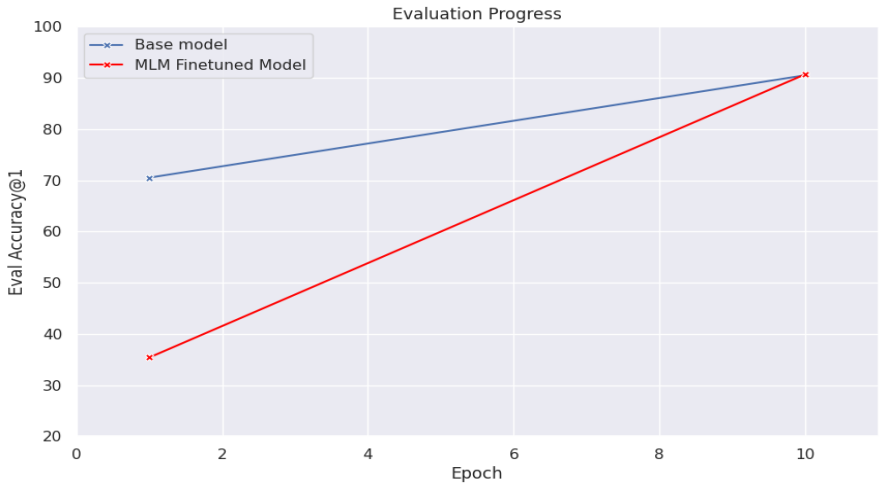

Figure 5.13: Progress and finale results of contrastive learning starting from MPNET base and MPNET with MLM pretrain on WHO-ICD9.

As it is possible to see in Fig 5.13, the final point reached by the two approaches is practically the same, however the starting point differs a lot. The MPNET base model without any further pretraining achieves a 70.5 in accuaracy@1 while the MPNET with additional pretrain only 35.37 in the same metric. Considering also the fact that both models after contrastive learning reach a peak of over 90.0 in accuaracy@1, it is possible to conclude that, as in MIMIC-III experiments 5.3.2, it is not the further pretraining that helps the model solve the task but rather the strategy adopted on the final task. In other words, while the MLM is not effective enough in enanching the capabilities on the specific semantic domain, the downstream strategy of contrastive learning is a suitable choice with respect to the multi-label classification.

Chapter 6

Discussion

This chapter summarizes all the conclusions drawn from the various experiments presented in chapter 5 and proposing some possible adjustments and changes to the TrustAlert project. As in the previous chapter, the considerations are divided for the two level of tagging.

6.1 Considerations over news filtering

The first level of tagging behaves very promisingly on real news from reputable sources such as the World Health Organisation and the BBC. This is promising considering that the real application of this news filtering stage will be specifically on the news collected by *GDELT*, which mainly resemble the blog posts and online newspapers from all over the world translated into English.

Another interesting but expected conclusion can be drawn from the model comparison. The *Sentence Transformers* such as *MPNET* are, due to their specific architecture, training strategies and data variety, the best choice for embedding long texts such as documents and news. Special attention should be paid to the pooling methods used to move from a token by token representation to a representation of the whole document.

Finally, the semantic enrichment of labels has some similarities with the behaviour of RAG systems. In fact, both rely on pre-trained language models to effectively understand and match semantic content. Therefore, some intuition from the RAG framework can be adapted to this pipeline, such as the use of the already implemented *FAISS*. Another portable solution may be contextualised label expansion and end-to-end training. That is, generating a label expansion via a generative model and training the generated expansion together with the retrieval task.

6.2 Considerations over ICD9 tagging

Considering the results achieved on tagging the news with the ICD9 taxonomy, it is possible to state that while on a news scenario the tagger behave quite effectively, on a more specific dataset that includes lots of specific terms and broken textual structure, such as MIMIC-III, the performances are far from optimal. This highlight mainly two problems: the necessity of quality data to work with in specific semantic scenarios and the difficulty of adapting a LLM on never seen vocabulary.

To address the first issue, a dataset was developed specifically for this thesis, but the need for golden truth for the TrustAlert project implies a total dataset length that is nowhere near the minimum amount typically used to train an LLM. However, the dataset using the World Health Organisation as the main source has the huge advantage of having ICD9 codes relevant to communicable diseases, whereas the most commonly used dataset in the literature, MIMIC-III, is clearly focused on more common health complications such as heart failure, which give TrustAlert almost no information to work with.

For the challenge of adapting some pre-trained LLMs to specific domains, the time and resource requirements remain a major obstacle. Therefor, an idea could be to explore data augmentation or refinement through purely synthetic LLM generation or textual paraphrasing. Another idea may be to use the generative capacity of LLM to generate label descriptions more useful for the retrieval task, or to generate a text conditioned by both code description and Golden Trough messages to have more sophisticated synthetic data.

However, given as a premise that the TrustAlert project will work on news, the technical terms and the broken structure of the text should not represent an obstacle for the developed tagger as shown in the last experiment proposed.

Another important consideration can be drawn from the comparison between models, again highlighting the effectiveness of a sentence transformer such as *MPNET*. An interesting fact is that if the same model is used for both levels of tagging, by simply switching the set of labels to retrieve, the computation time can be almost halved. In fact, if the backbone remains the same, the news embedding can be generated only once and not once for each level of tagging.

Finally, using contrastive learning as a framework and the flexible structure of the tagger architecture, it is possible to keep the model constantly updated through a cycle of online training, allowing continuous learning and adaptation, making the model more responsive to recent data trends and changes. Interesting as this is, it also raises again the issue of data quality.

Chapter 7

Conclusion

To conclude this thesis, I think it is necessary to recapitulate the main values of this work and to highlight some possible future work that can be done on the basis of this thesis.

7.1 Thesis Recap

This thesis has been conducted within the framework of the LINKS Foundation's contribution to the TrustAlert project, a collaborative initiative aimed at enhancing the early detection of healthcare crises.

The primary objective of this work was to identify the best method for classifying news streams from *GDELT* in a completely unsupervised scenario and to develop a robust test suite for evaluating the performance of the classification model. Given the dynamic nature of healthcare crises and the unsupervised nature of the available data, the research prioritized the exploration of models capable of zero-shot learning and flexible architectures.

Key contributions of this thesis include:

- **Development of a Flexible Tagger**: The creation of a versatile tagging system capable of classifying any text with any set of labels using the zero-shot capabilities of a pre-trained backbone model. This tagger leverages the strengths of Large Language Models (LLMs) to adapt to various classification tasks without the need for extensive fine-tuning.

- **Creation of a Specialized Dataset**: A dataset was developed to quantitatively evaluate the performance of the tagger on texts that mimic the format and style of the news articles the model is intended to filter. This ensures that the model's performance metrics are reflective of real-world application scenarios.

- **Comparison of Zero-Shot and Fine-Tuned Models**: An in-depth comparison was conducted between the zero-shot application of LLM models and their fine-tuning for specific domains, such as ICD9 code classification. This comparison provided insights into the trade-offs and benefits of each approach.
- **Investigation of Contrastive Learning**: The research explored contrastive learning techniques to fine-tune LLM models for classification tasks. This method aimed to enhance the models' performance without compromising their inherent zero-shot capabilities, offering a balanced approach to model adaptation and generalization.

The findings and developments presented in this thesis contribute significantly to the TrustAlert project by providing advanced tools for early anomaly detection in healthcare data and news streams. The flexible tagger and the evaluation dataset developed herein are valuable assets for ongoing and future research, facilitating the continuous improvement and application of machine learning models in dynamic and unsupervised environments. Through these efforts, the thesis underscores the importance of innovative approaches in tackling real-world challenges and advancing the field of healthcare crisis management.

7.2 Future Work

The findings and developments of this thesis have laid a strong foundation for the ongoing efforts within the TrustAlert project. However, there are several areas for future research that can further enhance the capabilities and effectiveness of the models developed. The following topics outline potential directions for future work:

- **Synthetic Data Generation for ICD Tagging**: A major challenge highlighted throughout this research was the scarcity of high-quality data for ICD tagging. To address this, future work should focus on the generation of synthetic data to augment the available datasets. By using advanced techniques such as *Generative Adversarial Networks* (GAN) or text generation models like *Generative Pretrained Transformers* (GPT) and any other *Large Language Model*, it is possible to create realistic and diverse synthetic samples that can help balance the dataset or generate samples for codes that do not even have any sample. This approach can significantly improve the training process, leading to more robust and accurate classification models.
- **Generation of expanded reliable datasets**: One of the main contributions of this thesis is the introduction of a reliable method to generate a dataset that is semantically matched to online news, with a focus on ICD9 communicable diseases. To further explore the possibilities of such a dataset, a viable

approach could be to use Google News and its metadata to retrieve and filter a specific subset of news items that talk about specific diseases.

- **Fine-Tuning Sentence Transformers with Vocabulary Expansion**: To improve the model's understanding of the semantics involved in both the classification and tagging tasks, future research should explore the fine-tuning of sentence transformers with vocabulary expansion. Vocabulary expansion is an idea driven by the fact that fine-tuned models very often retain the same vocabulary as the pre-trained model. This is because it makes no sense to add new words to a vocabulary mapped to random vectors, when all the other tokens have been trained over several epochs and their order is correlated with their frequency in the training corpus. Efforts should therefore be made to find a meaningful vector initialisation for these injected terms. However, by extending the vocabulary of the model to include domain-specific terms, the model should better capture the nuances and context of the input data. Fine-tuning with an expanded vocabulary may lead to improved performance in understanding and classifying complex domain-specific text.
- **Geometric Learning and Knowledge Graphs for Semantic Modeling**: A promising avenue for future work is represented by the study and application of geometric deep learning techniques. By leveraging geometric deep learning techniques and *Graph Neural Networks* (GNN), it is possible to model the relationships within the taxonomy in a semantic latent space through a knowledge graph that represent the hierarchical and relational structure of the taxonomy itself. The goal is to enable the model to understand and enforce these relationships during classification. This approach can improve the model's ability to capture the inherent structure and dependencies within the data, leading to more accurate and semantically consistent classifications.

These future research directions have the potential to significantly advance the TrustsAlert project. By addressing current limitations and working with larger amounts of high quality data, semantically aware models, and pushing into new, untested architectures, it is certainly possible to improve the overall performance of the TrustAlert project. These efforts will contribute to the development of more robust, accurate and semantically aware systems for early detection and management of health crises.

Bibliography

[1] Zellig S. Harris. «Distributional Structure». In: *WORD* 10.2-3 (1954), pp. 146–162. DOI: 10.1080/00437956.1954.11659520 (cit. on p. 7).

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 (cit. on p. 7).

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. «A neural probabilistic language model». In: *J. Mach. Learn. Res.* 3.null (Mar. 2003), pp. 1137–1155. ISSN: 1532-4435 (cit. on p. 7).

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. «Enriching Word Vectors with Subword Information». In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146. ISSN: 2307-387X (cit. on p. 8).

[5] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. «Bag of Tricks for Efficient Text Classification». In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Apr. 2017, pp. 427–431 (cit. on p. 8).

[6] Rico Sennrich, Barry Haddow, and Alexandra Birch. *Neural Machine Translation of Rare Words with Subword Units*. 2016. arXiv: 1508.07909 (cit. on pp. 8, 19).

[7] Minh-Thang Luong and Christopher D. Manning. *Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models*. 2016. arXiv: 1604.00788 (cit. on p. 8).

[8] Yonghui Wu et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. arXiv: 1609.08144 (cit. on pp. 8, 19).

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 (cit. on pp. 8, 16).

[10] Jeffrey L. Elman. «Finding structure in time». In: *Cognitive Science* 14.2 (1990), pp. 179–211. ISSN: 0364-0213. DOI: https://doi.org/10.1016/0364-0213(90)90002-E. URL: https://www.sciencedirect.com/science/article/pii/036402139090002E (cit. on p. 9).

[11] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-term Memory». In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735 (cit. on p. 10).

[12] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. *Deep contextualized word representations*. 2018. arXiv: 1802.05365 (cit. on p. 11).

[13] Holger Schwenk. «Continuous space language models». In: *Computer Speech & Language* 21.3 (2007), pp. 492–518. ISSN: 0885-2308 (cit. on p. 12).

[14] Holger Schwenk. «Continuous Space Language Models For Statistical Machine Translation». In: *The Prague Bulletin of Mathematical Linguistics* 93 (Feb. 2010), pp. 137–146 (cit. on p. 12).

[15] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078 (cit. on p. 13).

[16] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078 (cit. on p. 13).

[17] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*. 2014. arXiv: 1409.1259 (cit. on p. 13).

[18] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 (cit. on p. 13).

[19] Yonghui Wu et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. arXiv: 1609.08144 (cit. on p. 13).

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is all you need». In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964 (cit. on pp. 13, 14).

[21] Alex Graves. *Generating Sequences With Recurrent Neural Networks*. 2014. arXiv: 1308.0850 (cit. on p. 14).

[22] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. *Convolutional Sequence to Sequence Learning*. 2017. arXiv: 1705.03122 (cit. on p. 15).

[23] Iz Beltagy, Matthew E. Peters, and Arman Cohan. *Longformer: The Long-Document Transformer*. 2020. arXiv: 2004.05150 (cit. on p. 15).

[24] Manzil Zaheer et al. *Big Bird: Transformers for Longer Sequences*. 2021. arXiv: 2007.14062 (cit. on p. 15).

[25] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context*. 2019. arXiv: 1901.02860 (cit. on p. 15).

[26] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. 2020. arXiv: 1906.08237 (cit. on pp. 15, 19–21).

[27] Alec Radford and Karthik Narasimhan. «Improving Language Understanding by Generative Pre-Training». In: 2018. URL: https://api.semanticscholar.org/CorpusID:49313245 (cit. on pp. 15, 16).

[28] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. *BERTScore: Evaluating Text Generation with BERT*. 2020. arXiv: 1904.09675 (cit. on p. 17).

[29] Nils Reimers and Iryna Gurevych. «Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks». In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992. DOI: 10.18653/v1/D19-1410. URL: https://aclanthology.org/D19-1410 (cit. on pp. 17, 58, 60).

[30] Jeffrey Pennington, Richard Socher, and Christopher Manning. «GloVe: Global Vectors for Word Representation». In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162 (cit. on p. 17).

[31] Florian Schroff, Dmitry Kalenichenko, and James Philbin. «FaceNet: A unified embedding for face recognition and clustering». In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015. DOI: 10.1109/cvpr.2015.7298682. URL: http://dx.doi.org/10.1109/CVPR.2015.7298682 (cit. on pp. 17, 58).

[32] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. «Language Models are Unsupervised Multitask Learners». In: 2019. URL: https://api.semanticscholar.org/CorpusID:160025533 (cit. on p. 18).

[33] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. «BioBERT: a pre-trained biomedical language representation model for biomedical text mining». In: *Bioinformatics* 36.4 (Sept. 2019). Ed. by Jonathan Wren, pp. 1234–1240. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btz682. URL: http://dx.doi.org/10.1093/bioinformatics/btz682 (cit. on pp. 18, 19).

[34] Iz Beltagy, Kyle Lo, and Arman Cohan. «SciBERT: A Pretrained Language Model for Scientific Text». In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Hong Kong, China: Association for Computational Linguistics, 2019, pp. 3615–3620. URL: https://aclanthology.org/D19-1371 (cit. on p. 19).

[35] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. arXiv: 1910.13461 (cit. on p. 19).

[36] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. *MPNet: Masked and Permuted Pre-training for Language Understanding*. 2020. arXiv: 2004.09297 (cit. on pp. 21–23).

[37] Congzheng Song, Shanghang Zhang, Najmeh Sadoughi, Pengtao Xie, and Eric P. Xing. «Generalized Zero-shot ICD Coding». In: *CoRR* abs/1909.13154 (2019). arXiv: 1909.13154. URL: http://arxiv.org/abs/1909.13154 (cit. on pp. 25, 33).

[38] Soham Gadgil, Derek Jow, and Anthony Y. Li. «Improvement of deep learning techniques for ICD-9 code classification using MIMIC-III medical notes». In: 2020. URL: https://api.semanticscholar.org/CorpusID:215541990 (cit. on pp. 25, 33).

[39] Zhichao Yang, Shufan Wang, Bhanu Pratap Singh Rawat, Avijit Mitra, and Hong Yu. *Knowledge Injected Prompt Based Fine-tuning for Multi-label Few-shot ICD Coding*. 2022. arXiv: 2210.03304 [cs.CL] (cit. on p. 25).

[40] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG] (cit. on p. 53).

[41] Ilya Loshchilov and Frank Hutter. «Fixing Weight Decay Regularization in Adam». In: *CoRR* abs/1711.05101 (2017). arXiv: 1711.05101. URL: http://arxiv.org/abs/1711.05101 (cit. on p. 53).

[42] Wenguang Yu, Yu Weng, Ronghua Lin, and Yong Tang. «CoSBERT: A Cosine-Based Siamese BERT-Networks Using for Semantic Textual Similarity». In: *Computer Supported Cooperative Work and Social Computing*. Ed. by Yuqing Sun, Tun Lu, Yinzhang Guo, Xiaoxia Song, Hongfei Fan, Dongning Liu, Liping Gao, and Bowen Du. Singapore: Springer Nature Singapore, 2023, pp. 376–389. ISBN: 978-981-99-2356-4 (cit. on p. 58).

[43] Chengcheng Han, Yuhe Wang, Yingnan Fu, Xiang Li, Minghui Qiu, Ming Gao, and Aoying Zhou. *Meta-Learning Siamese Network for Few-Shot Text Classification*. 2023. arXiv: 2302.03507 [cs.CL] (cit. on p. 58).

[44] R. Hadsell, S. Chopra, and Y. LeCun. «Dimensionality Reduction by Learning an Invariant Mapping». In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. 2006, pp. 1735–1742 (cit. on p. 58).

[45] Alexandre Audibert, Aurélien Gauffre, and Massih-Reza Amini. *Exploring Contrastive Learning for Long-Tailed Multi-Label Text Classification*. 2024. arXiv: 2404.08720 [cs.LG] (cit. on p. 58).

[46] Daniela N. Rim, DongNyeong Heo, and Heeyoul Choi. «Adversarial Training with Contrastive Learning in NLP». In: *CoRR* abs/2109.09075 (2021). arXiv: 2109.09075. URL: https://arxiv.org/abs/2109.09075 (cit. on p. 58).

[47] Tianyu Gao, Xingcheng Yao, and Danqi Chen. «SimCSE: Simple Contrastive Learning of Sentence Embeddings». In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, pp. 6894–6910. DOI: 10.18653/v1/2021.emnlp-main.552. URL: https://aclanthology.org/2021.emnlp-main.552 (cit. on p. 58).

[48] R. Hadsell, S. Chopra, and Y. LeCun. «Dimensionality Reduction by Learning an Invariant Mapping». In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. 2006, pp. 1735–1742 (cit. on p. 58).

[49] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R. Scott. *Multi-Similarity Loss with General Pair Weighting for Deep Metric Learning.* 2020. arXiv: 1904.06627 (cit. on p. 58).